Collusion Resistant Copy-Protection for Watermarkable Functionalities

Jiahui Liu[*] Qipeng Liu[†] Luowen Qian[‡] Mark Zhandry[§]

Abstract

Copy-protection is the task of encoding a program into a quantum state to prevent illegal duplications. A line of recent works studied copy-protection schemes under "$1 \to 2$ attacks": the adversary receiving one program copy cannot produce two valid copies. However, under most circumstances, vendors need to sell more than one copy of a program and still ensure that no duplicates can be generated. In this work, we initiate the study of collusion resistant copy-protection in the plain model. Our results are twofold:

- The feasibility of copy-protecting all watermarkable functionalities is an open question raised by Aaronson et al. (CRYPTO' 21) In the literature, watermarking decryption, digital signature schemes and PRFs have been extensively studied.
  For the first time, we show that digital signature schemes can be copy-protected. Together with the previous work on copy-protection of decryption and PRFs by Coladangelo et al. (CRYPTO' 21), it suggests that many watermarkable functionalities can be copy-protected, partially answering the above open question by Aaronson et al.
- We make all the above schemes (copy-protection of decryption, digital signatures and PRFs) k bounded collusion resistant for any polynomial k, giving the first bounded collusion resistant copy-protection for various functionalities in the plain model.

1 Introduction

The idea of exploiting the quantum no-cloning principle for building cryptography was pioneered by Wiesner. In his seminal work [Wie83], he proposed the notion of quantum banknotes that cannot be counterfeited due to the unclonability of quantum information. This idea has profoundly influenced quantum cryptography, for example, inspiring the famous work on secure quantum key exchange [BB84]. Since all classical information is inherently clonable, unclonable cryptography is only achievable through the power of quantum information.

Aaronson [Aar09] further leveraged the capability of no-cloning to achieve copy-protection. The idea of copy-protection is the following. A software vendor wants to sell a piece of software, abstracted as a classical function f. It prepares a quantum state ρ_f so that anyone with a copy of ρ_f can evaluate f on a polynomial number of inputs. However, no efficient pirate receiving a single copy of ρ_f, could produce two programs that compute f correctly.

[*]University of Texas at Austin. Email: jiahui@cs.utexas.edu
[†]Simons Institute for the Theory of Computing. Email: qipengliu0@gmail.com
[‡]Boston University. Supported by DARPA under Agreement No. HR00112020023. Email: luowenq@bu.edu
[§]Princeton University & NTT Research. Email: mzhandry@gmail.com

The notion above intuitively captures the security of a copy-protection scheme under what we call an "$1 \to 2$ attack": the adversary receives 1 program copy, and attempts to produce 2 copies with the correct functionality. A recent line of works [ALL+21, CLLZ21, CMP20, AKL+22] achieve secure copy-protection for various functionalities under $1 \to 2$ attacks.

However, such a security notion is extremely limiting: in most circumstances, we cannot expect the software vendor to issue only one copy of the program. When the vendor gives out multiple copies, all users can collude and generate pirate copies together. Therefore, a useful copy-protection scheme should be secure against any "$k \to k+1$ attack" for any polynomial k. Such security is usually referred to as collusion resistance in the literature.

Prior Works on Copy-Protection We first recall on a high level how most existing copy-protection schemes work: a copy-protection program consists of a quantum state as an "unclonable token", and a classical part containing an obfuscated program (either as an oracle or the output coming out of some obfuscation functionality). The obfuscated program takes in a token and an input one requests to evaluate on; it verifies the validity of the token and if the verification passes, it outputs the evaluation on the requested input.[1]

Until now, collusion resistant copy-protection has essentially been wide open. The only work that considers issuing more than a single program is Aaronson's original work [Aar09], which is proven to be secure in the $k \to k+r$ setting for $r \geq k$ in some structured quantum oracle model. This is undesirable in two ways: (a) it is unclear whether the scheme allows an adversary to double the copies of programs (Aaronson leaves improving r as a challenging open question), which is not a complete break but still potentially devastating to applications; but more importantly, (b) unlike a classical oracle which could be heuristically instantiated using indistinguishability obfuscation, we do not even know how to heuristically instantiate a quantum oracle. Moreover, we believe that any extension of Aaronson's scheme would very likely still require some obfuscation of quantum circuits, since we have evidence that Haar random states, which is the core of Aaronson's scheme, lack the structure that can be verified by a classical circuit [Kre21].

If we turn to the other works constructing copy-protection without using quantum oracles, one naïve idea is to take any such scheme that is $1 \to 2$ secure, and simply generate and hand out multiple copies of ρ_f. It turns out that while this satisfies correctness, they are all trivially broken once two copies are given. This is because they are all based on quantum states that are unclonable for one copy, but trivially clonable as soon as two copies are given.

To get around this issue, another idea is to instead employ a quantum state that already bears a "$(k \to k+1)$-unclonable" property. However, the only known such states are Haar random states and its computationally (or statistically) close neighbors, such as pseudorandom states (or t-designs), which leads us back to the verification issue without a quantum oracle from before.

Therefore, we raise the natural question: *Is collusion resistant copy-protection feasible, either resisting $k \to k+1$ attacks, or without using a quantum oracle? (Ideally both?)*

Copy-Protection in the Plain Model In this work, we restrict our attention to investigate the question above in the plain model, i.e. we want provably secure protocols without any oracle or heuristics. Unfortunately, it has been known that copy-protection in the plain model even

[1]The *general functionality* copy protection schemes in [Aar09, ALL+21] and the schemes in [CLLZ21, AP21] all satisfy this format. The copy-protection schemes for *point/compute-and-compare functions* in [Aar09, CMP20, AKL+22, BJL+21] are not necessarily of such a format.

for unlearnable functions is impossible in general [AP21], and thus we have to further restrict ourselves to construct copy-protection for specific classes of functions that evade the impossibility.

Secure software leasing (SSL) [AP21] is a weakened notion for copy-protection: in (infinite-term) SSL, the malicious pirate may attempt to make pirate copies as it wants. However, the freeloaders are restricted to running a fixed public quantum circuit on some quantum state produced by the pirate. On the other hand, in copy-protection, the freeloaders are free to execute any quantum circuit that the pirate asks them to. Despite facing the same impossibility as copy-protection, secure software leasing has also been built for various functionalities [AP21, CMP20, BJL$^+$21, ALL$^+$21, KNY21]. [2]

Especially, [ALL$^+$21, KNY21] showed that secure software leasing for watermarkable functions could be obtained from watermarking and public key quantum money in a black-box way. Watermarking [BGI$^+$01] is a primitive that embeds a watermark into a program so that any attempt to remove the watermark would destroy the program's functionality. Observing this, Aaronson et al. [ALL$^+$21] raised the following open question: *Can all watermarkable functions also be copy-protected in the plain model?*

In this work, we will use the word "major watermarkable functions" to denote (decrypting) public key encryption, (signing) signatures, and (evaluating) PRFs and only focus on copy-protecting those functionalities. Starting from the work by Cohen et al. [CHN$^+$18], a line of works [KW17, GKM$^+$19, KW19, YAL$^+$19, YAYX20] focuses on watermarking these three functionalities. Copy-protecting these cryptographic functionalities also has a natural and strong motivation: the ability to evaluate these functions is supposedly private in many circumstances. If owners of a decryption key, signing key, or PRF key can share their key with others, it will trigger severe security concerns. Furthermore, copy-protecting a cryptographic function can lead to copy-protecting a software entity of which this cryptographic function is a component.

We observe that collusion resistant secure software leasing for watermarkable functions can be achieved as long as the underlying watermarking scheme and quantum money scheme are both collusion resistant, by looking into the construction in [ALL$^+$21, KNY21]. (Bounded) collusion resistant watermarking for PRFs, public-key encryptions, etc. are constructed in the plain model [GKM$^+$19, YAL$^+$19, YAYX20, ...] and quantum money can be made collusion resistant with a digital signature on its serial number [AC13]. This observation seems to suggest that collusion resistant copy-protection could be a much more challenging goal.

1.1 Our Results

In this work, we partially answer all of the questions above. In particular, we show how, in the plain model, to construct collusion resistant copy-protection for (decrypting) public-key encryption, (signing) signatures, and (evaluating) PRFs. Our results, together with the prior work on copy-protection of decryption and PRFs (Coladangelo et al. [CLLZ21]), show that major watermarkable cryptographic functionalities can be copy-protected against even colluding adversaries, in the plain model. We now explain this in more detail.

Collusion Resistant Unclonable Decryption Our first result is collusion resistant copy-protection for decryption keys in a public-key encryption scheme. We refer to such copy-protection scheme

[2]The formal security definitions for SSL in [AP21, CMP20, BJL$^+$21, ALL$^+$21, KNY21] vary slightly from one to another. We will discuss them in 1.2.

as *unclonable decryption* by convention, as first proposed by Georgiou and Zhandry [GZ20].

Theorem 1.1. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and subexponentially secure LWE, there exists k-bounded collusion resistant unclonable decryption for any polynomial k.*

Our collusion resistant unclonable decryption scheme is based on the construction from the prior work of Coladangelo et al. [CLLZ21] that achieves the same except with only $1 \to 2$ security. Note that while we require subexponential security, these assumptions match those already required in the prior work. In particular, here, we invoke subexponential security only for a compute-and-compare obfuscation scheme with certain properties as our building block. All the reductions in this work are polynomial.

While we do achieve $k \to k+1$ security, a caveat is that we only achieve "k-bounded collusion resistance", by which we mean that we need a preset number of users k to generate the public key. Still, we consider *all* users as potentially malicious and colluding. We note that this is similar to watermarking decryption circuits of public-key encryption schemes, where to the best of our knowledge, unbounded collusion resistance is also unknown [YAL$^+$19, GKM$^+$19]. Furthermore, it is foreseeable that bounded collusion resistance suffices in certain enterprise use cases where the number of (partially) authorized parties is a priori known and fixed; furthermore, such tokens can be transferred to a new employee irrevocably.

The main challenges are in the anti-piracy security proof. The prior proof idea for $1 \to 2$ anti-piracy does not translate to the $k \to k+1$ setting. We present a new view on security reductions to handle a polynomial number of possibly entangled quantum adversaries, which we will elaborate in the technical overview.

Copy-Protecting Watermarkable Functionalities We complement the previous theorem regarding public-key encryption, with the following result on collusion resistant copy-protection for signatures and PRFs:

Theorem 1.2. *Assuming post-quantum subexponentially secure indistinguishability obfuscation and subexponentially secure LWE, there exists k-bounded collusion resistant copy-protection for digital signatures and PRFs, for any polynomial k.*

We base our construction on the signature token scheme and unclonable PRF in the plain model built in [CLLZ21] (with $1 \to 2$ anti-piracy). However, our signature scheme is significantly different in two aspects: (a) the signing key in [CLLZ21] will be consumed after one use whereas our scheme is reusable, and (b) unforgeability breaks down when multiple signature queries can be issued, whereas ours satisfies standard existential unforgeability.

1.2 Related Works

[Aar09] first built copy-protection for all unlearnable functions based on a quantum oracle, with weak collusion resistance. Besides [CLLZ21] which we have discussed, [ALL$^+$21] showed a construction for all unlearnable functions based on a classical oracle. [CMP20, AKL$^+$22] constructed copy-protection for point functions and compute-and-compare functions in QROM, the latter improving the security of the former. [3]

[3] All constructions discussed in this section are not proved under collusion resistant security unless otherwise specified.

Regarding the negative results: [AP21] demonstrated that it is impossible to have a copy-protection scheme for all unlearnable circuits in the plain model, assuming LWE and quantum FHE. [AK22] extended this impossibility result to the setting where we allow approximate correctess of the copy-protection program and working in the classical-accessible random oracle model.

[AP21] put forward secure software leasing (SSL). In the finite-term case, a software vendor would lease a quantum state as the software to a user; later, the user needs to return a part of a bipartite state to the vendor, and the vendor will use its own secret key to verify if this returned state is the one issued in the authentic program. The security guarantees that while passing the above verification, the user should not be able to evaluate the functionality correctly using the other part of its bipartite state executed under a public, fixed quantum circuit eval (specified by the vendor). In the infinite-term case, the user does not need to return the state to the vendor; the security guarantees that it should not produce two states that can both evaluate the function correctly when executed under eval. [AP21] also built an (infinite-term) SSL scheme for searchable compute-and-compare circuits under iO and LWE.

[ALL$^+$21] observed that under a definition essentially equivalent to infinite-term SSL, namely copy-detection, one could obtain a black-box construction for infinite-term SSL from watermarking and public-key quantum money. [KNY21] constructed finite-term SSL for PRFs and compute-and-compare functions from (subexponential) LWE, with similar observations.

[BJL$^+$21, CMP20] constructed secure software leasing for point functions and compute-and-compare functions; [BJL$^+$21] is information-theoretically secure and [CMP20] is secure under QROM. They both used a stronger version of finite-term SSL security: while the vendor will honestly check the returned state from the adversary, the adversary can execute the leftover half of its bipartite state maliciously, i.e., not following the instructions in eval. SSL security of this stronger finite-term variant is only known for point/compute-and-compare functions up till now.

1.3 Technical Overview

We start by showing how to overcome the aforementioned barriers and construct Collusion Resistant Unclonable Decryption (CRUD). As briefly discussed in the introduction, there are challenges to constructing collusion resistant copy-protection based on the so-called "$k \to (k+1)$ no-cloning theorem". Instead, we take a different approach by constructing collusion resistant unclonable decryption CRUD from unclonable decryption UD whose security only holds for "$1 \to 2$ attacks". The construction uses UD in a black-box manner:

- For every $i \in [k]$, sample $(|\mathsf{sk}_i\rangle, \mathsf{pk}_i) \leftarrow \mathsf{UD.KeyGen}$; $|\mathsf{sk}_i\rangle$ will be the i-th copy of the quantum unclonable decryption key; the public key will be $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$.
- The encryption algorithm takes a single bit message m and outputs a classical ciphertext ct that consists of k copies of ciphertext, among which the i-th copy ct_i is the ciphertext of m under pk_i.
- To decrypt $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_k)$ with $|\mathsf{sk}_i\rangle$, one can decrypt the i-th ciphertext ct_i.

Intuitively in the above encryption scheme, one can decrypt only if it knows the decryption key for at least one of the public keys. Note that our k decryption keys are sampled independently at random and each state satisfies $1 \to 2$ unclonability. To establish anti-piracy, we want to prove a security reduction from a $k \to k+1$ quantum pirate decryptors to the $1 \to 2$ unclonability of one of the decryption keys.

Unfortunately, we do not know how to prove the security of this scheme generically. As we will elaborate in Section 1.4, we need to open up the construction of the underlying unclonable encryption in order to establish the security.

More importantly, in the following section, we demonstrate that even if we open up the construction and the proof, the proof technique in [CLLZ21] seems not sufficient for CRUD and we thereby work on a new technique that subsumes that in [CLLZ21] to complete the proof. We start by recalling the definition of regular UD and the proof in [CLLZ21].

Regular Unclonable Decryption. Let UD be a regular $(1 \to 2)$ unclonable decryption scheme. For the sake of convenience, we assume the message space is $\{0, 1\}$. A pair of a classical public key pk and a quantum unclonable secret key $|\mathsf{sk}\rangle$ is generated by KeyGen.

The anti-piracy security guarantees that no efficient adversary with $|\mathsf{sk}\rangle$ can produce two "working" keys by a CPA indistinguishability standard: if one estimates the success probabilities of both decryption keys on distinguishing a ciphertext of 0 from a ciphertext of 1, their success probabilities cannot be *simultaneously* significantly greater than $1/2$, except with negligible probability. This security notion has been previously studied by Aaronson et al. [ALL+21] and Coladangelo et al. [CLLZ21]

Before we delve into the security proof, it is enlightening to see how this security guarantee is efficiently "falsifiable". Estimating the success probability of a *classical* decryptor is easy. One can generate a ciphertext for a random message using the public key and check whether the classical decryptor is correct on that ciphertext; then, a simple counting estimates its success probability within any inverse polynomial error. Unfortunately, this method does not naturally work in the quantum setting since a single execution of the decryption key (produced by the adversary) may disturb the state and prevent further execution of the same key.

Nevertheless, Zhandry [Zha20] shows that such estimation can be done analogous to the classical setting, inspired by the famous work of Marriot and Watrous [MW05] for witness-preserving error reduction for quantum Arthur–Merlin game. Informally, the work of Zhandry utilizes a measurement procedure called "projective implementation" (abbreviated as PI)[4] to estimate the success probability of a quantum adversary (see Figure 1).

1. Let $\mathcal{D}$ be a ciphertext distribution we define the procedure with respect to.
2. For any quantum decryptor σ with success probability p over $\mathcal{D}$, running $\mathsf{PI}_{\mathcal{D}}$ on the decryptor produces a probability p' and σ collapses to σ';
3. σ' as a decryptor, has success probability p' over $\mathcal{D}$;
4. Applying $\mathsf{PI}_{\mathcal{D}}$ on σ' always produces p' and σ' remains intact;
5. The expectation of p' is p.

$$(\sigma, p) \longrightarrow \boxed{\mathsf{PI}_{\mathcal{D}}} \longrightarrow (\sigma', p') \longrightarrow \boxed{\mathsf{PI}_{\mathcal{D}}} \longrightarrow (\sigma', p')$$
$$\mathbb{E}[p'] = p$$

Figure 1: PI: measure success probability of a decryptor.

[4]For simplicity, we only use the inefficient estimation procedure. The same argument in the technical overview holds using an efficient and approximated version. Similarly for TI.

Put shortly, this measurement procedure will output an estimation of the success probability p' for a quantum decryptor σ. After the measurement, the decryptor collapsed to another decryptor σ', whose success probability is still p'. We will intuitively call PI as "probability estimation' instead of its original name in the scope of the overview.

In the anti-piracy security definition, we care about whether both decryptors have the success probability significantly greater than $1/2$. [CLLZ21] defines the following "threshold measurement" or "goodness measurement" $\mathsf{TI}_{\mathcal{D},\epsilon}$ for deciding if a quantum decryptor σ is good, for some inverse-polynomial ϵ:

1. Let $\mathcal{D}$ be a ciphertext distribution we define the procedure with respect to.
2. Run $\mathsf{PI}_{\mathcal{D}}$ coherently on σ and measure if the outcome register (containing the resulting probability p') is greater than $1/2 + \epsilon$, which produces a single bit outcome b. The quantum decryptor collapses to σ'.
3. If $b = 1$, σ' lies in the span of good decryptors, whose success probability is at least $1/2 + \epsilon$; otherwise, σ' is in the subspace with the basis being quantum decryptors whose winning probability is strictly less than $1/2 + \epsilon$.

$$\begin{tikzcd}\sigma \arrow[r] & \boxed{\mathsf{TI}_{\mathcal{D},\epsilon}} \arrow[r] & (\sigma', b)\end{tikzcd}$$

Figure 2: TI: measure goodness of a decryptor.

We note that $\mathsf{TI}_{\mathcal{D},\epsilon}$ is a projection, which says if σ' is the collapsed decryptor for outcome b, applying $\mathsf{TI}_{\mathcal{D},\epsilon}$ will always produce b and σ' does not change.

We are now ready to formally define the anti-piracy security in [CLLZ21]. Let $\mathcal{D}$ be the ciphertext distribution for honestly generated ciphertext, which encodes a uniformly random message. No efficient adversary can turn $|\mathsf{sk}\rangle$ into a possibly entangled decryptors σ over two registers, such that applying the threshold measurement $\mathsf{TI}_{\mathcal{D},\epsilon}$ on both decryptors $\sigma[1], \sigma[2]$ will produce two outcomes 1s with non-negligible probability. To put it another way, no efficient adversary can produce two decryptors such that they jointly have non-negligible weight on good decryptors.

Security Proof for "$1 \to 2$ Attacks". Before scoping the proof of our collusion resistant unclonable decryption, we recall the security proof in [CLLZ21] for "$1 \to 2$ unclonability". In this following section, we will highlight the difficulties of applying the same ideas to CRUD and introduce a new approach to resolve this issue.

The proof works as follows:

- A reduction applies $\mathsf{TI}_{\mathcal{D},\epsilon}$ on both decryptors $\sigma[1], \sigma[2]$. With some non-negligible probability, it will produce two outcomes 1s and the two decryptors become $\sigma'[1], \sigma'[2]$.
- **Extraction on the first register.** Let $\mathcal{D}'$ be the ciphertext distribution for "junk" ciphertext which only encrypts an empty symbol $\perp$. Applying $\mathsf{TI}_{\mathcal{D}',\epsilon}$ on $\sigma'[1]$ always result in outcome 0, whereas the outcome of applying $\mathsf{TI}_{\mathcal{D},\epsilon}$ on $\sigma'[1]$ is always 1.
  We can thereby conclude that $\sigma'[1]$ must contain some secret information about the secret key $|\mathsf{sk}\rangle$. In fact, we can use an extraction algorithm to extract the classical information about the secret key. Note that the algorithm may be destructive that, for example, may measure $\sigma'[1]$ completely.

- **Extraction on the second register.** Conditioned on the successful extraction on $\sigma'[1]$, we want to argue that a similar extraction on the second register works. If so, we can *simultaneously* extract secret information about $|\mathsf{sk}\rangle$ from two non-communicating parties. This will violate the underlying quantum information guarantee[5].

The remaining is to show such an extraction is feasible on the second decryptor, even conditioned on the successful extraction on $\sigma'[1]$. This is because $\mathsf{TI}_{\mathcal{D},\epsilon}$ is a projection, conditioned on the outcome being 1, $\sigma'[2]$ will be in the span of good decryptors (see bullet (3) of the description of TI). Regardless of what event is conditioned on $\sigma'[1]$, the second decryptor is still in the span of good decryptors. Thus, an extraction algorithm would extract the classical information about the secret key from $\sigma'[2]$ with non-negligible probability. This concludes the proof idea in [CLLZ21].

To conclude, the core idea in the proof is that, a "$1 \to 2$ attack" produces two quantum registers that

1. they have a non-negligible probability $w_1 = \gamma$ on both registers being good decryptors on $\mathcal{D}$ (with success probabilities at least $1/2 + \epsilon$);
2. they have a negligible probability w_2 on both being good decryptors on $\mathcal{D}'$.

If both 1 and 2 are satisfied, a simultaneous extraction succeeds with a non-negligible probability.

In the next few paragraphs, we still denote w_1 as the joint probability of both decryptors being good on distribution $\mathcal{D}$; w_2 as the joint probability of both decryptors being good on distribution $\mathcal{D}'$.

In the above proof for $1 \to 2$ attack, we crucially require w_1 is non-negligible and w_2 is negligible or zero, in order to argue that extraction would succeed even after conditioned on successful extraction on one side.

We can also observe that for the $1 \to 2$ proof, w_2 is automatically zero. As $\mathcal{D}'$ does not encode a real message, no quantum decryptor can achieve any advantage over random guessing. But this is *not* always the case when it turns to our CRUD security proof: for which, $\mathcal{D} = (\mathsf{ct}_\perp, \cdots, \mathsf{ct}_j, \mathsf{ct}_{j+1}, \cdots)$ has the first $(j-1)$ ciphertexts being junk and the rest being real; whereas $\mathcal{D}' = (\mathsf{ct}_\perp, \cdots, \mathsf{ct}_\perp, \mathsf{ct}_{j+1}, \cdots)$ has the first j ciphertexts being junk.

As we will see in the following section, for CRUD, the condition "$w_1 - w_2$ is non-negligible" is the best we can hope for. Therefore, we attempted to see if a proof similar to the above exists, when we can only condition on "$w_1 - w_2$ is non-negligible". Unfortunately, the answer to this attempt is negative, as we will provide some intuition in the immediate next paragraph. We thereby conclude that the proof technique in [CLLZ21] cannot extend to collusion resistant anti-piracy security proof in a generic way.

To see why the condition "$w_1 - w_2$ is non-negligible" does not necessarily give a simultaneous extraction, we consider the time when a successful extraction has already been done on the first decryptor $\sigma'[1]$. If w_2 is negligible, the leftover state of the second decryptor $\sigma'[2]$ has at most w_2/ζ weight lying in the span of bad decryptors. Here ζ is the probability of a successful extraction on the first decryptor and conditioned on this extraction, the weight w_2 will be amplied by at most $1/\zeta$. Since w_2/ζ is still negligible, this allows an extraction from $\sigma'[2]$ happens with a non-negligible chance. However, if w_2 is not negligible but only satisfies $w_1 - w_2$ is non-negligible, $\sigma'[2]$

[5]In the actual proof, two non-communicating parties will extract two vectors, one in the primal coset and the other in the dual coset of a coset state. This will violate the strong computational monogamy-of-entanglement property of coset states.

can lie in the span of bad decryptors: the extreme case will be the event of successful extraction on $\sigma'[1]$ has "positive correlation" with $\sigma'[2]$ being bad; in this case, the weight can be as large as $w_2/\zeta \approx 1$.

Obstacles for Extraction from Quantum Decryptors. The high-level intuition for why such a construction would satisfy $k \to k+1$ is comprehensible. Assume an adversary uses $|\mathsf{sk}_1\rangle, \cdots, |\mathsf{sk}_k\rangle$ to produce $(k+1)$ (possibly entangled) malicious decryptors σ. Let $\sigma[i]$ denote the i-th pirate decryptor. Since each $\sigma[i]$ is a "working" pirate decryptor, it should at least decrypt one of $\mathsf{ct}_1, \cdots, \mathsf{ct}_k$ (say ct_j). Applying pigeonhole principle, there are two decryptors that decrypts the same ciphertext slot, which would violate $1 \to 2$ unclonability. However, such an intuition is nontrivial to formalize since a quantum adversary could distribute these secret keys in multiple ways in superposition.

A straightforward idea is to extract secret information for the j-th private key $|\mathsf{sk}_j\rangle$ from $\sigma[i]$. Let $\mathcal{D}'$ be the ciphertext distribution $(\mathsf{ct}_\perp, \mathsf{ct}_\perp, \cdots, \mathsf{ct}_\perp)$ containing all junk ciphertext. Clearly, if we apply $\mathsf{TI}_{\mathcal{D}',\epsilon}$ on any quantum decryptor, the result is always 0 (meaning "bad"). If we can find an index j such that $\mathcal{D}_j$ is the distribution $(\mathsf{ct}_\perp, \mathsf{ct}_\perp, \cdots, \mathsf{ct}_j, \cdots, \mathsf{ct}_\perp)$ and applying $\mathsf{TI}_{\mathcal{D}_j,\epsilon}$ on $\sigma[i]$ gives 1 with non-negligible chance, we can extract secrets for $|\mathsf{sk}_j\rangle$ from $\sigma[i]$. If one can extract from every $\sigma[i]$, by the pigeonhole principle, it breaks the underlying quantum information guarantee for one of the unclonable decryption keys. Unfortunately, this idea does not go through, considering the following bad situation.

Even if $\sigma[i]$ has success probability 1, such j may not exist. Consider a quantum program that knows all the decryption keys $|\mathsf{sk}_1\rangle, \cdots, |\mathsf{sk}_k\rangle$ but only decrypts ct if and only if every $|\mathsf{sk}_j\rangle$ can successfully decrypt ct_j; if any decryption fails to decrypt, it outputs a random guess. Feeding $(\cdots, \mathsf{ct}_\perp, \mathsf{ct}_j, \mathsf{ct}_\perp, \cdots,)$ to the decryptor will always result in a random guessing.

Note that this is not only an issue for quantum decryptors but also presents if decryptors are classical. A natural fix of the above idea is to consider the following hybrid distributions. We define $\mathcal{D}_j$ for every $j \in \{0, 1, \cdots, k\}$:

- $\mathcal{D}_j$: = $(\mathsf{ct}_\perp, \cdots, \mathsf{ct}_\perp, \mathsf{ct}_j, \mathsf{ct}_{j+1} \cdots)$. In other words, only the last $k-j$ ciphertexts encode the same random message $m \in \{0,1\}$, the first j ciphertexts are junk ciphertexts .
- $\mathsf{TI}_j := \mathsf{TI}_{\mathcal{D}_j,\epsilon}$: the goodness estimation with respect to the ciphertext distribution $\mathcal{D}_j$ and threshold $1/2+\epsilon$.

That is, each $\mathcal{D}_j$ will replace the first non-junk ciphertext from $\mathcal{D}_{j-1}$ with a junk ciphertext. Note that $\mathcal{D} := \mathcal{D}_0$. By the definition of $\sigma[i]$ is a working decryptor, applying TI_0 on $\sigma[i]$ will produce 1 with a non-negligible probability. On the flip side, applying TI_k on $\sigma[i]$ will always produce 0.

We denote w_j as the probability of applying $\mathsf{TI}_{\mathcal{D}_j,\epsilon}$ on the decryptor $\sigma[i]$ and getting outcome 1. By a standard hybrid argument, we can conclude that there must exist an index $j \in [k]$ such that,

$$w_{j-1} - w_j \text{ is non-negligible.}$$

The gap allows extraction on $\sigma[i]$. However, as we discussed in the last section, it does not satisfy the condition "w_{j-1} is non-negligible and w_j is negligible", which can not guarantee a simultaneous extraction when we consider two decryptors.

A bad example looks like the following: $w_0 = \gamma$ for some inverse polynomial γ and $w_j = \gamma/2^j$ for all $j \neq k$ and $w_k = 0$. There does not exists a j such that w_{j-1} is non-negligible but w_j is negligible.

We now elaborate on our approaches to resolve these obstacles. Our approach directly takes advantage of the probability measure PI instead of TI. This also gives an alternative security proof for the construction in [CLLZ21].

Extract a Single Decryption Key: Detect a Large Jump in Success Probability Let us start with attempts to extract from a single "working" decryptor σ, using the probability estimation PI. Recall that by the definition of "working", we mean applying $\mathsf{PI}_{\mathcal{D}}$ on σ yields some probability p significantly larger than the trivial guessing probability $1/2$.

We first recall the following ciphertext distributions $\mathcal{D}_j$ and define probability estimation procedure PI_j for every $j \in \{0, 1, \cdots, k\}$:

- $\mathcal{D}_j$: = $(\mathsf{ct}_\perp, \cdots, \mathsf{ct}_\perp, \mathsf{ct}_j, \mathsf{ct}_{j+1} \cdots)$.
- $\mathsf{PI}_j := \mathsf{PI}_{\mathcal{D}_j}$: the probability estimation with respect to the ciphertext distribution $\mathcal{D}_j$.

Now we give the following attempted extraction, which almost works but has one caveat. We call this extraction procedure a "repeated probability estimation/measurement":

1. We first apply PI_0 to σ and obtain p_0 and a collapsed decryption key σ_0.
2. We then apply PI_1 to the collapsed σ_0 to obtain p_1 and σ_1.

   Now if $p_1 - p_0$ is at least $\frac{p_0 - \frac{1}{2}}{k}$, we perform an extraction procedure to extract secrets for $|\mathsf{sk}_1\rangle$ from σ_0. Intuitively, since we observe a noticeable probability decrease when ct_1 is replaced with junk ciphertext, there must be some part of $\sigma[i]$ that uses ct_1 to recover the original plaintext. We then abort the procedure.
3. Otherwise, p_0 and p_1 should be negligibly close. We again apply PI_2 on σ_1 and obtain p_2, σ_2.

   If $p_2 - p_1$ is at least $\frac{p_0 - \frac{1}{2}}{k}$, we perform extraction on σ_1 and abort.
4. We continue this process for all $j = 3, ..., k$.

We claim that the above repeated measurement procedure will always terminate at some $j \in [k]$. To see this, think of $p_1, ..., p_k$ as a sequence of random variables, whose values are only observed when the corresponding measurement is applied. Note that $p_k = 1/2$ always, because the underlying ciphertext distribution $\mathcal{D}_k$ encodes all junk ciphertexts, so no adversary can achieve better advantage than guessing. Therefore, the claim follows from triangle inequality.

$$(\sigma_0, p_0) \xrightarrow{\mathsf{PI}_1} (\sigma_1, p_1) \xrightarrow{\mathsf{PI}_2} (\sigma_2, p_2) \xrightarrow{\mathsf{PI}_3} \cdots \xrightarrow{\mathsf{PI}_{k-1}} (\sigma_k, p_k)$$

The above extraction procedure almost works. But it is actually not physically executable: we need σ_{j-1} in order to perform extraction as that is the state with a "working" component for ciphertext ct_j, but by the time that we decide to extract, we already get to state σ_j because we have to obtain measurement outcome p_j to claim a jump in probability happens. It is generally infeasible to rewind a quantum state, in this case from σ_j to σ_{j-1}.[6]

[6]The probability estimation PI_j will preserve the success probability of the state but nothing else. Applying PI_j will likely change σ_{j-1}.

Fortunately, it is plausible for a single decryptor: we guess j (denoting the first index having a probability jump) and stop the procedure when we have done $\mathsf{PI}_0, \cdots, \mathsf{PI}_{j-1}$. With probability at least $1/k$, we can extract for $|\mathsf{sk}_j\rangle$ from the current decryptor σ_{j-1}. We will get to why this procedure avoids the rewinding issue and preserves our success probability, when it comes to the $(k+1)$ decryptors case in the next paragraph.

Extending to $(k+1)$ **decryptors.** Finally, we show how to generalize the above extraction strategy to extracting secrets from the same key $|\mathsf{sk}_j\rangle$.

We apply the repeated measurement individually to every decryptor: that is, for the i-th decryptor, we apply $\mathsf{PI}_0, \mathsf{PI}_1, \cdots, \mathsf{PI}_k$, one upon another. The procedure will yield $p_{i,0}, p_{i,1}, \cdots, p_{i,k}$. Since $p_{i,0}$ is always greater than $1/2 + \gamma$ and $p_{i,k} = 1/2$, there must exist a large probability gap between p_{i,j_i-1} and p_{i,j_i} for some $j_i \in [k]$. By the pigeonhole principle, for some $x \neq y$, $j := j_x = j_y$. We hope to stop at the x-th and y-th decryptors before applying PI_j and simultaneously turn them into two keys for ct_j.

Since there will always be two decryptors having large probability gaps for the same index, the chance of having such gaps for randomly guessed x, y and j is at least $\frac{1}{\binom{k+1}{2}k} \geq 1/k^3$. But the success probability of this guess is not immediately guaranteed, because we need to stop before the j-th probability estimation for states $\sigma[x], \sigma[y]$ otherwise we can't rewind to this state needed for extraction. We are still two unpredictable measurements away from the event we guess for. Fortunately, guessing and stopping before the j-th PI will indeed work with probability at least $1/(2k^3)$, through a trick for randomized algorithms.

Now we can apply repeated measurement and stop before applying PI_j on any of these two decryptors. Let the leftover decryptors be $\sigma^*[x, y]$ and the last probability outcomes be $p_{x,j-1}$ and $p_{y,j-1}$. With probability at least $1/(2k^3)$, $(\sigma^*[x, y], p_{x,j-1}, p_{y,j-1})$ satisfy the following conditions (*) and (**):

(*) Applying PI_{j-1} on both $\sigma^*[x]$ and $\sigma^*[y]$ always produces $p_{x,j-1}$ and $p_{y,j-1}$.

(**) Applying PI_j on both $\sigma^*[x]$ and $\sigma^*[y]$, with probability at least $1/(2k^3)$, produces large probabilities gaps for both $p_{x,j-1}$ and $p_{y,j-1}$.

It seems that we have come to the right "spot" for extraction. However, we still face a challenge. How do we guarantee that we can simultaneously extract from two possibly entangled states? A possible malicious behavior is that measuring one decryptor's key will collapse the other decryptor to a "not working" state.

We can clearly extract secrets for $|\mathsf{sk}_j\rangle$ from either $\sigma^*[x]$ or $\sigma^*[y]$: since there is a probability gap, it must mean $\sigma^*[x]$ (or $\sigma^*[y]$) use ct_j for decryption at some point. From the probability point of view, we then argue why simultaneous extraction is feasible.

Define $\mathbf{E}_x$ ($\mathbf{E}_y$, here $\mathbf{E}$ stands for "(E)xtraction") be the event of a successful extraction on the x-th decryptor (or on the y-th decryptor respectively). Define $\mathbf{G}_x$ ($\mathbf{G}_y$, here $\mathbf{G}$ stands for "(G)ap") be the event that applying PI_j on the x-th decryptor (or on the y-th decryptor respectively) yields a large probability gap. We will prove $\Pr[\mathbf{E}_x \wedge \mathbf{E}_y]$ is non-negligible by contradiction.

It is clear that $\Pr[\mathbf{E}_x]$ is non-negligible. To show $\Pr[\mathbf{E}_y|\mathbf{E}_x]$ is non-negligible, it is sufficient to show that $\Pr[\mathbf{G}_y|\mathbf{E}_x]$ is non-negligible, since a large gap implies a large chance of extraction.

We can intuitively think of $\Pr[\mathbf{E}_x] = 0.1 \Pr[\mathbf{G}_x]$ and $\Pr[\mathbf{E}_y] = 0.1 \Pr[\mathbf{G}_y]$[7]. We may expect

[7]The choice of 0.1 is arbitrary here. Indeed, they are polynomially related. For the sake of simplicity, we assume they are linearly related.

that $\Pr[\mathbf{E}_x \wedge \mathbf{E}_y] = 0.1 \Pr[\mathbf{G}_x \wedge \mathbf{G}_y]$, which would conclude the proof. However, this does not follow immediately from above as it could be the case that $\mathbf{G}_x \wedge \mathbf{G}_y$ occurs with non-negligible probability, but $\mathbf{E}_x \wedge \mathbf{E}_y$ never occurs. The main insight here is that we can instead show that $\Pr[\mathbf{E}_x|\mathbf{G}_y] = 0.1 \Pr[\mathbf{G}_x|\mathbf{G}_y]$, as finding the gap for y does not impact the extraction for x. Invoking Bayes' rule, this shows that $\Pr[\mathbf{G}_y|\mathbf{E}_x] = \Pr[\mathbf{E}_x|\mathbf{G}_y] \Pr[\mathbf{G}_y] / \Pr[\mathbf{E}_x]$ is non-negligible as well. As a consequence, $\Pr[\mathbf{E}_y|\mathbf{E}_x]$ and thus $\Pr[\mathbf{E}_x \wedge \mathbf{E}_y]$ (simultaneous extraction) are both large.

Collusion Resistant Copy-Protection for Signatures and PRFs Now with the building block of collusion resistant unclonable decryption, we come to copy-protect more cryptographic functions.

As briefly discussed in the introduction, even though [CLLZ21] presented the first unclonable signature scheme without oracles, its scheme is a signature token that will be consumed after one use. One-time signature is a security notion interesting under many circumstances [BS16, GZ20], but it's crucial that we investigate the possibility of copy-protecting a standard digital signature. Moreover, once achieved, this construction helps us get closer to the goal of copy-protecting all watermarkable functionalities.

The [CLLZ21] signature token is one-time because when signing a message, the signer simply measures the quantum key and the measurement outcome is a signature. It is not existentially unforgeable for the same reason: if an adversary gets a few random measurement results of quantum keys, he is granted the power to sign, without the need of an intact quantum key.

To resolve the problem, we resort to the classic picture of generic copy-protection: the signing program first verifies if a quantum key is a valid "token" and then outputs a signature (computed independently of the quantum key) as well as the almost unharmed key. In particular, we observe that the unclonable decryption scheme in [CLLZ21] will pave the way for such a construction. Their scheme can be extended to a copy-protection for evaluating puncturable PRFs with the " hidden trigger" technique from [SW21]. Meanwhile, such PRF evaluation functionality can be used as a signing program after obfuscation.

We thereby give a copy-protection for existentially unforgeable, publicly-verifiable signature scheme, based on the above ideas. Along the way, we deal with a few subtleties that emerge because we need public verification and generalization to collusion resistance. More specifically, we present a k-party version of the [SW21] hidden trigger technique to obtain both collusion resistant copy-protection for signatures and for PRFs.

1.4 Discussions and Open Problems

Comparisons to [CLLZ21]. An informed reader may claim that one main obstacle (namely simultaneous extraction) for proving anti-piracy security in this paper resembles the obstacle in the $1 \to 2$ anti-piracy schemes of [ALL+21, CLLZ21]. We emphasize that while this issue may be bumped into in all quantum copy-protection proofs, our approach of resolving the issue is different from previous works, especially to identify gaps in a repeated probability estimation procedure (see more details in the technical overview). In particular, our approach can be used to prove security for the schemes in [ALL+21] and [CLLZ21], but as we have discussed in the technical overview, their techniques will *not* work for the $k \to k+1$ setting [8]

[8]The approach for simultaneous extraction when showing $1 \to 2$ anti-piracy in [ALL+21] bears a high-level similarity with [CLLZ21]. We have discussed [CLLZ21] in the overview since we focus on unclonable decryption.

On Non-Black-Box Reduction. In the technical overview, we describe a black-box way of reducing "$k \to (k+1)$ security" to "$1 \to 2$ security". As mentioned earlier, we cheat in the technical overview and the approach is not entirely black-box.

A high-level summary for the reason is: a black-box reduction algorithm (i.e. an adversary for a $1 \to 2$ unclonable decryption scheme) is not able to generate the correct distribution for the ciphertext to feed to the k collusion resistant adversary. Elaborated as follows:

First, recall that in a k collusion resistant scheme, an encryption for a message m is an ensemble of ciphertexts $\mathsf{ct} = (\mathsf{ct}_1, ..., \mathsf{ct}_k)$ where $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}_i, m)$ for all $i \in [k]$.

In the reduction, we want to apply $\mathsf{PI}_{\mathcal{D}_j}$ on a malicious decryptor to extract secrets from $|\mathsf{sk}_j\rangle$ for some $j \in [k]$:

$\mathcal{D}_j$: the first j ciphertexts (that is, $\mathsf{ct}_1, \cdots$ up to ct_j) are simulated ciphertexts, the rest of them encrypt the same message.

The problem is the following: the reduction only gets a single ciphertext ct_{j+1}, whereas the malicious decryptor takes input of the form in $\mathcal{D}_j$. The reduction needs to generate other ciphertext on its own: including those simulated and those encrypting the same message as c_{j+1}. Since the reduction does not know which message is encrypted in ct_{j+1} (otherwise, the reduction itself already breaks the security of the underlying $1 \to 2$ unclonable decryption), it cannot generate a valid ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_k)$ from the distribution $\mathcal{D}_j$.

Therefore, we need to open this proof up in a non-black-box way: it's based on the security of coset states. When we break the security of coset states, the message (encrypted in ct_{j+1}) is known by the reduction. In fact, it is even sampled by the reduction R.

Open Problems. The main limitation of our constructions is that the number of collusions is bounded to a polynomial specified during setup, and the parameters grow with the collusion bound. Because of this collusion bound, our results are technically incomparable to [Aar09], which, despite having a much weaker copy-protection guarantee and using a strong oracle, required no prefixed user number. We leave achieving unbounded $k \to k+1$ collusion resistance as an interesting open question.

1.5 Organization

The rest of the paper is organized as follows. In Section 2, we recall the definitions and properties of coset states and how to measure success probabilities of quantum adversaries. In Section 3, we present the definition, construction, and security proof of collusion resistant unclonable decryption. We then present the construction and definitons for the copy-protection for signatures in 4. The constructions and security proofs for (collusion resistant) copy-protection for signature schemes and PRFs are covered in the appendix.

2 Preliminaries

In this paper, λ denotes the security parameter. $\mathsf{poly}(\cdot)$ denotes a polynomial function. We say a function $f(\cdot) : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is negligible if for all constant $c > 0$, $f(n) \leq \frac{1}{n^c}$ for all sufficiently large n. $\mathsf{negl}(\cdot)$ denotes a negligible function. Similarly, we say a function $f(\cdot) : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is sub-exponential if there exists a constant $c < 1$, such that $f(n) \leq 2^{n^c}$ for all sufficiently large n.

$\mathsf{subexp}(\cdot)$ denotes a sub-exponential function. For an integer k, We denote $\{1, 2, \cdots, k\}$ by $[k]$. We denote $\mathbb{F}_2$ to be the binary field.

We refer the reader to [NC10] for a reference of basic quantum information and computation concepts.

2.1 Indistinguishability Obfuscation

Definition 2.1 (Indistinguishability Obfuscator (iO) [BGI+01, GGH+16, SW21]). *A uniform PPT machine* iO *is an indistinguishability obfuscator for* $\mathsf{P/poly}$ *if the following conditions are satisfied:*

- *For all λ, all $|C| \leq \lambda$, all inputs x, we have*
$$\Pr\Big[\widehat{C}(x) = C(x) \;:\; \widehat{C} \leftarrow \mathsf{iO}(1^\lambda, C)\Big] = 1.$$
- *(Post-quantum security): For all (not necessarily uniform) QPT adversaries (Samp, D), the following holds: if $\Pr[\forall x, C_0(x) = C_1(x) \wedge |C_0| = |C_1| \;:\; (C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)] > 1 - \alpha(\lambda)$ for some negligible function α, then there exists a negligible function β such that:*
$$\Big|\Pr\Big[D(\sigma, \mathsf{iO}(1^\lambda, C_0)) = 1\Big] - \Pr\Big[D(\sigma, \mathsf{iO}(1^\lambda, C_1)) = 1\Big]\Big| \leq \beta(\lambda),$$
where $(C_0, C_1, \sigma) \leftarrow \mathsf{Samp}(1^\lambda)$.

The notion *sub-exponentially secure* iO denotes an indistinguishability obfuscator, for which no QPT adversary can achieve advantage better than $1/\mathsf{subexp}$ for some sub-exponential function subexp.

2.2 Coset States

We recall the notion of coset states, introduced by [VZ21] and later studied by [CLLZ21] in the setting of quantum copy-protection. We then present a property of coset states: a strong computational monogamy-of-entanglement (MOE) property. This property is used to obtain an unclonable decryption scheme and other copy-protection of watermarkable cryptographic primitives in this work. Some part of this section is taken verbatim from [CLLZ21].

2.2.1 Definitions

For any subspace A, its complement is $A^\perp = \{b \in \mathbb{F}_2^n \,|\, \langle a, b\rangle = 0\,, \forall a \in A\}$. It satisfies $\dim(A) + \dim(A^\perp) = n$. We also let $|A| = 2^{\dim(A)}$ denote the number of elements in the subspace A.

Definition 2.2 (Coset States). *For any subspace $A \subseteq \mathbb{F}_2^n$ and vectors $s, s' \in \mathbb{F}_2^n$, the coset state $|A_{s,s'}\rangle$ is defined as:*
$$|A_{s,s'}\rangle = \frac{1}{\sqrt{|A|}} \sum_{a \in A} (-1)^{\langle s', a\rangle} |a + s\rangle \ .$$

By applying $H^{\otimes n}$ (Hadamard on every qubit) on the state $|A_{s,s'}\rangle$, one obtains exactly $|A^\perp_{s',s}\rangle$. Given A, s and s', there is an efficient quantum algorithm that generates $|A_{s,s'}\rangle$, by [CLLZ21].

For a subspace A and vectors s, s', we define cosets $A + s = \{v + s : v \in A\}$, and $A^\perp + s' = \{v + s' : v \in A^\perp\}$. It is also convenient for later sections to define a canonical representative, with respect to subspace A, of the coset $A + s$.

Definition 2.3 (Canonical Representative of a Coset). *For a subspace A, we define the function $\mathsf{Can}_A(\cdot)$ such that $\mathsf{Can}_A(s)$ is the lexicographically smallest vector contained in $A + s$ (we call this the canonical representative of coset $A + s$).*

[CLLZ21] showed that, Can_A and $\mathsf{Can}_{A^\perp}$ are efficiently computable given the classical description of A.

When it is clear from the context, we will write $A + s$ to denote the *program* that checks membership in $A + s$. The following equivalences, which follow straightforwardly from the security of iO, will be useful in our security proofs later on.

Proposition 2.4. *For any subspace $A \subseteq \mathbb{F}_2^n$, $\mathsf{iO}(A+s) \approx_c \mathsf{iO}(\mathsf{CC}[\mathsf{Can}_A, \mathsf{Can}_A(s)])$. Recall that $\mathsf{CC}[\mathsf{Can}_A, \mathsf{Can}_A(s)]$ refers to the compute-and-compare program which on input x outputs 1 if and only if $\mathsf{Can}_A(x) = \mathsf{Can}_A(s)$.*

This is due to the fact that $A + s$ has the same functionality as $\mathsf{CC}[\mathsf{Can}_A, \mathsf{Can}_A(s)]$. The lemma then follows the security of iO.

2.2.2 Strong Monogamy-of-Entanglement Property

Consider a game between a challenger and an adversary $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$:

- The challenger picks a uniformly random subspace $A \subseteq \mathbb{F}_2^n$ of dimension $\frac{n}{2}$, and two uniformly random elements $s, s' \in \mathbb{F}_2^n$. It sends $|A_{s,s'}\rangle$, $\mathsf{iO}(A + s)$, and $\mathsf{iO}(A^\perp + s')$ to $\mathcal{A}_0$.
- $\mathcal{A}_0$ creates a bipartite state on registers B and C. Then, $\mathcal{A}_0$ sends register B to $\mathcal{A}_1$, and C to $\mathcal{A}_2$.
- The classical description of A is then sent to both $\mathcal{A}_1, \mathcal{A}_2$.
- $\mathcal{A}_1$ and $\mathcal{A}_2$ return respectively s_1 and s_2.

$(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ wins if and only if $s_1 \in A + s$ and $s_2 \in A^\perp + s'$.

Let $\mathsf{CompStrongMonogamy}((\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2), n)$ be a random variable which takes the value 1 if the game above is won by adversary $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, and takes the value 0 otherwise.

Theorem 2.5. *Assuming the existence of sub-exponentially secure post-quantum iO and one-way functions, then for any QPT adversary $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$,*

$$\Pr[\mathsf{CompStrongMonogamy}((\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2), n) = 1] \leq 1/\mathsf{subexp}(n)\,.$$

[CV21] proved an information-theoretic version of the strong monogamy property (without giving out the iO programs to the adversary). [CLLZ21] showed that one can obtain the computational statement by lifting the information-theoretic statement.

2.3 Measure Success Probabilities of Quantum Adversaries: Projective/Threshold Implementation

In this section, we include several definitions and results about estimating success probabilities or estimating whether the probability is above a threshold. Part of this section is taken verbatim from [ALL+21, CLLZ21]. In this section, we will mainly talk about how to measure probability in an inefficient way. The proofs in the main body of the proof use this inefficient measuring procedure as subroutines. All these proofs can be translated easily using the efficient version of such measuring procedures. We will cover those in the appendix.

Estimating success probabilities of adversaries is essential in many settings, especially for a reduction to know whether the adversary is good or if an extraction on the adversary can succeed with high probability. Classically it is easy. Let $\mathcal{D}$ be a testing input distribution and C be a classical program for which we want to estimate probability. We can keep running C on uniformly fresh inputs sampled from $\mathcal{D}$ to estimate the probability up to any inverse polynomial error. Such procedure is infeasible for quantum adversaries, since a single execution of a quantum program may completely collapse the program, leading to failure for future executions.

Projective Implementation Zhandry [Zha20] formalizes the following probability measurement procedure for a quantum program ρ under some test distribution $\mathcal{D}$.

Consider the following procedure as a binary POVM $\mathcal{P}_{\mathcal{D}} = (P_{\mathcal{D}}, Q_{\mathcal{D}})$ acting on a quantum program ρ (whose success probability is equal to p): sample an input x from $\mathcal{D}$, evaluates the quantum program ρ on x, and checks if the output is correct. Let $P_{\mathcal{D}}$ denote the operator for output being correct and $Q_{\mathcal{D}}$ be the quantum operator for the output being incorrect.

Zhandry proposed a procedure that applies an appropriate projective measurement which *measures* the success probability of ρ on input $x \leftarrow \mathcal{D}$, and outputs the probability p'. Conditioned on the outcome is some probability p', the quantum program collapsed to ρ' whose success probability is exactly p'. Furthermore, the expectation of p' equals to p.

Theorem 2.6 (Projective Implementation)**.** *Let $\mathcal{D}$ be a distribution of inputs. Let $\mathcal{P}_{\mathcal{D}} = (P_{\mathcal{D}}, Q_{\mathcal{D}})$ be a binary outcome POVM described above with respect to the distribution $\mathcal{D}$. There exists a projective measurement $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ such that for any quantum program ρ with success probability p on $\mathcal{D}$:*

(i) Applying $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ on ρ yields ρ', p'.
(ii) ρ' has success probability p' with respect to $\mathcal{D}$. Furthermore, applying $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ on ρ' always produces p'.
(iii) The expectation of p' equals to p.

We say the above measurement procedure is a projective implementation of $\mathcal{P}_{\mathcal{D}}$. When the distribution is clear from the context, we sometimes ignore the subscript $\mathcal{D}$ in both $\mathcal{P}_{\mathcal{D}}$ and $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$.

Threshold Implementation The concept of threshold implementation [ALL+21] is similar to projective implementation, except it now outputs a binary outcome indicating whether the probability is above or below some threshold.

Theorem 2.7 (Threshold Implementation)**.** *Let $\mathcal{D}$ be a distribution of inputs. Let $\mathcal{P}_{\mathcal{D}} = (P_{\mathcal{D}}, Q_{\mathcal{D}})$ be a binary outcome POVM described above with respect to the distribution $\mathcal{D}$. For any $0 \leq \gamma \leq 1$, there exists a projective measurement $\mathsf{TI}_\gamma(\mathcal{P}_{\mathcal{D}})$ such that for any quantum program ρ:*

(i) Applying $\mathsf{TI}_\gamma(\mathcal{P}_{\mathcal{D}})$ on ρ yields a binary outcome b' and a collapsed program ρ'.
(ii) If $b' = 1$, ρ' has success probability at least γ with respect to $\mathcal{D}$. Furthermore, applying $\mathsf{TI}_\gamma(\mathcal{P}_{\mathcal{D}})$ on ρ' always produces 1.
(iii) If $b' = 0$, ρ' has success probability less than γ with respect to $\mathcal{D}$. Furthermore, applying $\mathsf{TI}_\gamma(\mathcal{P}_{\mathcal{D}})$ on ρ' always produces 0.

We say the above measurement procedure is a threshold implementation of $\mathcal{P}_{\mathcal{D}}$ with threshold γ. When the distribution is clear from the context, we sometimes ignore the subscript $\mathcal{D}$ in $\mathsf{TI}(\mathcal{P}_{\mathcal{D}})$.

Moreover, $\mathsf{TI}(\mathcal{P}_{\mathcal{D}})$ can be implemented by first applying $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ to get a outcome p and outputting 1 if $p \geq \gamma$ or 0 otherwise.

For simplicity, we denote by $\mathrm{Tr}[\mathsf{TI}_\gamma(\mathcal{P}_\mathcal{D})\,\rho]$ the probability that the threshold implementation applied to ρ **outputs 1**. Thus, whenever $\mathsf{TI}_\gamma(\mathcal{P}_\mathcal{D})$ appears inside a trace Tr, we treat $\mathsf{TI}_\gamma(\mathcal{P}_\mathcal{D})$ as a projection onto the 1 outcome.

The approximate and efficient versions of both PI and TI will be covered in the Appendix A.2.

3 Collusion Resistant Unclonable Decryption

In this section, we give the formal definition of collusion resistant unclonable decryption. We will then show the construction for achieving bounded collusion resistance for any k — polynomial number of parties. Finally, we prove the construction satisfies correctness, semantic security and anti-piracy against colluding adversaries. Our scheme has security against bounded number of parties. It requires to know the parameter k in the setup phase and only k copies of keys can be generated later. Furthermore, the public key, secret key and ciphertext have length linear in the number of parties k. Note that our scheme is secure even if an adversary takes control of all copies of decryption keys; the adversary still can not produce any additional functioning key.

3.1 Definitions

Definition 3.1 (Bounded Collusion Resistant Unclonable Decryption Scheme)**.** *A bounded collusion resistant unclonable decryption scheme* CRUD *for a message space* $\mathcal{M}$ *consists of the following efficient algorithms:*

- $\mathsf{Setup}(1^\lambda, k) \to (\mathsf{sk}, \mathsf{pk})$: *a (classical) probabilistic polynomial-time (in λ, k) algorithm that takes as input an upper bound k on the number of users and a security parameter λ and outputs a classical secret key* sk *and a classical public key* pk*.*
- $\mathsf{QKeyGen}(\mathsf{sk}) \to \rho_{\mathsf{sk},1} \otimes \rho_{\mathsf{sk},2} \otimes \cdots \otimes \rho_{\mathsf{sk},k}$: *a quantum algorithm that takes as input a secret key* sk *and outputs k copies of quantum secret keys.*
- $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$: *a (classical) probabilistic algorithm that takes as input a public key* pk*, a message m and outputs a classical ciphertext* ct*.*
- $\mathsf{Dec}(\rho_{\mathsf{sk}}, \mathsf{ct}) \to m/\bot$: *a quantum algorithm that takes as input a quantum secret key ρ_{sk} and a classical ciphertext* ct*, and outputs a message m or a decryption failure symbol $\bot$.*

Here 'bounded' refers to the restriction that the Setup procedure requires to know the maximal number of keys distributed in the QKeyGen.

A bounded collusion resistant unclonable decryption scheme should satisfy the following:

Correctness: For every polynomial $k(\cdot)$, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $\lambda \in \mathbb{N}$, let $k := k(\lambda)$, for all $m \in \mathcal{M}$, all $i \in [k]$,

$$\Pr\left[\mathsf{Dec}(\rho_{\mathsf{sk},i}, \mathsf{ct}) = m \;\middle|\; \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda, k), \\ \rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

In other words, correctness says the i-th quantum decryption key will always decrypt correctly (except with negligible probability). By the gentle measurement lemma [Aar05], each decryption key can function correctly polynomially many times for honestly generated encryptions.

CPA Security: This is the regular semantic security for an encryption scheme. An adversary without getting any decryption key (neither sk nor these quantum keys) can not distinguish ciphertexts of chosen plaintexts.

Formally, for every (stateful) QPT adversary $\mathcal{A}$, for every polynomial $k(\cdot)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\Pr\left[\mathcal{A}(\mathsf{ct}) = b : \begin{array}{c} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda, k) \\ ((m_0, m_1) \in \mathcal{M}^2) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk}) \\ b \leftarrow \{0,1\}; \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

Anti-Piracy Security Finally, we define anti-piracy against colluding adversaries. Anti-piracy intuitively says there is no adversary who gets all copies of the decryption keys can successfully produce one additional "working" key.

We will follow the two different definitions of "working" proposed in [CLLZ21] and give two definitions for anti-piracy. The first definition allows a pirate to announce two messages (m_0, m_1), much like the semantic security. A decryption key is good if an adversary can distinguish encryptions of m_0 and m_1 by using the decryption key. The second definition of a "working" decryption key is basing on whether it decrypts correctly with high probability on uniformly random inputs.

Before describing the security games, we first recall the concept of a quantum decryptor (or a quantum decryption key) [CLLZ21] with respect to a collusion resistant unclonable decryption scheme.

Definition 3.2 (Quantum Decryptor). *A quantum decryptor ρ for ciphertexts of length m, is an ℓ-qubit state for some polynomial ℓ. For a ciphertext c of length m, we say that we run the quantum decryptor ρ on ciphertext c to mean that we execute a universal quantum circuit U on inputs $|c\rangle$ and ρ, and measure the output registers.*

We are now ready to describe the CPA-style anti-piracy game as well as the random challenge anti-piracy game. We first introduce the notion of good decryptors with respect to two messages (m_0, m_1).

Definition 3.3 (($\frac{1}{2} + \gamma$)-good Test with respect to (m_0, m_1)). *Let $\gamma \in [0, 1/2]$. Let pk be a public key, and (m_0, m_1) be a pair of messages. We refer to the following procedure as a test for a γ-good quantum decryptor with respect to pk and (m_0, m_1):*

- *The procedure takes as input a quantum decryptor ρ.*
- *Let $\mathcal{P} = (P, I - P)$ be the following POVM acting on some quantum state ρ':*
  - *Sample a uniform $b \leftarrow \{0,1\}$ and random coins r. Compute $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b; r)$.*
  - *Run the quantum decryptor on input c. Check whether the outcome is m_b. If so, output 1; otherwise output 0.*
- *Let $(\mathsf{TI}_{1/2+\gamma}, I - \mathsf{TI}_{1/2+\gamma})$ be the threshold implementation of $\mathcal{P}$ with threshold value $\frac{1}{2} + \gamma$, as defined in Theorem 2.7. Run the threshold implementation on ρ, and output the outcome. If the output is 1, we say that the test passed, otherwise the test failed.*

Definition 3.4 (k-Strong-Anti-Piracy Game, CPA-style). *Let $\lambda, k \in \mathbb{N}^+$. The CPA-style strong anti-piracy game for a collusion resistant unclonable decryption scheme is the following game between a challenger and an adversary $\mathcal{A}$.*

1. ***Setup Phase***: *The challenger samples keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Setup}(1^\lambda, k)$*.*
2. ***Quantum Key Generation Phase***: *The challenger sends* $\mathcal{A}$ *the classical public key* pk *and all* k *copies of quantum decryption keys* $\rho = \rho_{\mathsf{sk},1} \otimes \cdots \rho_{\mathsf{sk},k} \leftarrow \mathsf{KeyGen}(\mathsf{sk})$*.*
3. ***Output Phase***: $\mathcal{A}$ *outputs a pair of distinct messages* (m_0, m_1)*. It also outputs a (possibly mixed and entangled) state* σ *over* $k+1$ *registers* $R_1, R_2, \cdots, R_{k+1}$*. We interpret* σ *as* $k+1$ *(possibly entangled) quantum decryptors* $\sigma[R_1], \cdots, \sigma[R_{k+1}]$*.*
4. ***Challenge Phase:*** *Let* $\mathsf{TI}_{1/2+\gamma}$ *be the* $(\frac{1}{2}+\gamma)$*-good test with respect to* (m_0, m_1)*. The challenger applies* $\mathsf{TI}_{1/2+\gamma}$ *to each of these decryptors. The challenger outputs* 1 *if and only if all the measurements output* 1*.*

We denote by $\mathsf{StrongAntiPiracyCPA}(1^\lambda, 1/2+\gamma, k, \mathcal{A})$ *a random variable for the output of the game.*

Definition 3.5 (Strong Anti-Piracy-Security)**.** *Let* $\gamma : \mathbb{N}^+ \to [0,1]$*. An unclonable decryption scheme satisfies strong* γ*-anti-piracy security, if for any polynomial* $k(\cdot)$*, for any QPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds for all* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[b = 1, b \leftarrow \mathsf{StrongAntiPiracyCPA}(1^\lambda, 1/2+\gamma(\lambda), k(\lambda), \mathcal{A})\right] \leq \mathsf{negl}(\lambda) \tag{1}$$

Note that the above strong anti-piracy security is defined by the threshold implementation TI. By [CLLZ21], this definition implies a weaker notion called *regular CPA-style anti-piracy security*, which says the probability of all $k+1$ malicious parties simultaneously distinguish encryptions of m_0 or m_1 (m_0 and m_1 are chosen independently for each malicious parties) is at most negligibly greater than $1/2$.

We can similarly define *regular anti-piracy security with random message challenges*: the probability of all $k+1$ malicious parties simultaneously recover ciphertext of independent random messages is at most negligibly greater than $1/2^n$, where n is the message length.

3.2 Construction

We now give the construction of our collusion resistant unclonable decryption. Let UD be the unclonable decryption scheme based on coset states [CLLZ21]. Our CRUD takes k as input and outputs k pairs of freshly generated keys for UD. A message is encrypted under each public key. Decryption works if a decryptor can decrypt any ciphertext. The construction of CRUD follows from the construction of UD. The security of our CRUD requires a non-black-box analysis for the last step.

We recall the unclonable decrytion scheme in [CLLZ21] (see Figure 4).

There is one additional function Sim which takes a parameter n (message length) and outputs a junk ciphertext, which will be crucial for our anti-piracy proof. Intuitively, if one can distinguish from a honestly generated ciphertext with a simulated ciphertext, they can extract secrets for the underlying coset states.

The efficiency, correctness and CPA security of our CRUD scheme follows easily from those of UD. We are focusing on the proof of its anti-piracy in the next section.

3.3 Proof of Anti-Piracy

In this section, we prove that our construction satisfies anti-piracy. Although the proof requires to open up the structure of UD, this only happens for the last step: for arguing we can extract secrets

$\mathsf{CRUD.Setup}(1^\lambda, k)$:

- For $i \in [k]$, $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{UD.Setup}(1^\lambda)$.
- Let $\mathsf{sk} = (\mathsf{sk}_1, \cdots, \mathsf{sk}_k)$ and $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$. Output $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{CRUD.QKeyGen}(\mathsf{sk})$:

- Parse $\mathsf{sk} = (\mathsf{sk}_1, \cdots, \mathsf{sk}_k)$. Let $\rho_i \leftarrow \mathsf{UD.QKeyGen}(\mathsf{sk}_i)$.
- Let $\rho_{\mathsf{sk},i}$ be ρ_i padded with a classical index i, i.e., $\rho_{\mathsf{sk},i} = \rho_i \otimes |i\rangle\langle i|$.
- Output $\rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k}$.

$\mathsf{CRUD.Enc}(\mathsf{pk}, m)$:

- Parse $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$. Let $\mathsf{ct}_i \leftarrow \mathsf{UD.Enc}(\mathsf{pk}_i, m)$.
- Output $\mathsf{ct}_1, \cdots, \mathsf{ct}_k$.

$\mathsf{CRUD.Dec}(\rho_{\mathsf{sk}}, \mathsf{ct})$:

- Parse $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_k)$. Parse ρ_{sk} as ρ and i.
- Output $\mathsf{UD.Dec}(\rho, \mathsf{ct}_i)$.

Figure 3: Collusion Resistant Unclonable Decryption.

for the underlying coset states using the properties of compute-and-compare obfuscation. Therefore, we will present the main idea of the proof here, leaving the proof of successful extraction (see Claim 3.11) in the appendix.

Theorem 3.6. *The construction in Section 3.2 has strong γ-anti-piracy for any inverse polynomial γ (as defined in Definition 3.5).*

Proof. We prove by contradiction. There exist inverse polynomials $\gamma(\cdot)$, $\nu(\cdot)$, $k(\cdot)$ and an adversary $\mathcal{A}$ such that for infinitely many $\lambda \in \mathbb{N}^+$, $\mathcal{A}$ outputs a pair of distinct messages (m_0, m_1) and a state σ over $k+1$ registers (which are $k+1$ decryptors) such that

$$\mathrm{Tr}\left[\left(\mathsf{TI}_{1/2+\gamma} \otimes \mathsf{TI}_{1/2+\gamma} \otimes \cdots \otimes \mathsf{TI}_{1/2+\gamma}\right)\sigma\right] \geq \nu. \tag{2}$$

Let σ^* be the leftover state (over the $k+1$ registers), conditioned on all $\mathsf{TI}_{1/2+\gamma}$ outputting 1. With Equation (2), we can get to σ^* with probability at least ν.

Next we will prove the theorem assuming we have perfect projective implementation (see below). Therefore, the resulting reduction is inefficient. At the end of the section, we will show the proof translates easily when we replace every projective implementation with its approximated and efficient version. This replacement will give us an efficient reduction and only incur a small loss.

Defining Probability Measurement PI. We start by defining the following measurements PI_i for each $i \in [k]$. PI_i stands for the projective implementation where the underlying ciphertext distribution is: the first i ciphertexts are "fake", without encoding any information about the plaintext; the rest are generated honestly. pk are $(\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$ as defined in our construction Section 3.2; similarly for sk_i.

$\mathsf{UD.Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$:

- Sample ℓ random $(\lambda/2)$-dimensional subspaces $A_i \subseteq \mathbb{F}_2^\lambda$ for $i = 1, 2, \cdots, \ell$, where $\ell := \ell(\lambda)$ is a polynomial in λ.
- For each $i \in [\ell]$, choose two uniformly random vectors $s_i, s_i' \in \mathbb{F}_2^n$.
- Prepare the programs $\mathsf{iO}(A_i + s_i)$ and $\mathsf{iO}(A_i^\perp + s_i')$ (where we assume that the programs $A_i + s_i$ and $A_i^\perp + s_i'$ are padded to some appropriate length).
- Output $\mathsf{sk} = \{A_i, s_i, s_i'\}_{i\in[\ell]}$, $\mathsf{pk} = \{\mathsf{iO}(A_i + s_i), \mathsf{iO}(A_i^\perp + s_i')\}_{i\in[\ell]}$.

$\mathsf{UD.KeyGen}(\mathsf{sk}) \to \rho_{\mathsf{sk}}$: on input $\mathsf{sk} = \{A_i, s_i, s_i'\}_{i\in[\ell]}$, output the "quantum secret key" $\rho_{\mathsf{sk}} = \{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell]}$.

$\mathsf{UD.Enc}(\mathsf{pk}, m) \to \mathsf{ct}$: on input a public key $\mathsf{pk} = \{\mathsf{iO}(A_i + s_i), \mathsf{iO}(A_i^\perp + s_i')\}_{i\in[\ell]}$ and message m:

- Sample a uniformly random string $r \leftarrow \{0,1\}^\ell$.
- Let r_i be the i-th bit of r. Define $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s_i')$. Let $\mathsf{P_{m,r}}$ be the following program Figure 5.
- Let $\hat{\mathsf{P}}_{m,r} = \mathsf{iO}(\mathsf{P_{m,r}})$. Output ciphertext $\mathsf{ct} = (\hat{\mathsf{P}}_{m,r}, r)$.

$\mathsf{UD.Dec}(\rho_{\mathsf{sk}}, \mathsf{ct}) \to m/\perp$: on input $\rho_{\mathsf{sk}} = \{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell]}$ and $\mathsf{ct} = (\hat{\mathsf{P}}_{m,r}, r)$:

- For each $i \in [\ell]$, if $r_i = 1$, apply $H^{\otimes n}$ to the i-th state $|A_{i,s_i,s_i'}\rangle$; if $r_i = 0$, leave the i-th state $|A_{i,s_i,s_i'}\rangle$ unchanged. Denote the resulting state by ρ_{sk}^*.
- Evaluate the program $\hat{\mathsf{P}}_{m,r}$ on input ρ_{sk}^* in superposition; measure the evaluation register and denote the outcome by m'. Output m'.
- Rewind by applying the operations in the first step again.

$\mathsf{UD.Sim}(n) \to \mathsf{ct}$: on input a message length n, $\mathsf{ct} \leftarrow \mathsf{iO}(\mathsf{Sim}(1^\lambda, P.\mathsf{param}))$ where Sim denotes the simulator for compute-and-compare obfuscator, $P.\mathsf{param}$ consists of all program parameters in $P_{m,r}$ as in $\mathsf{UD.Enc}$ for any m of length n.

Figure 4: Unclonable Decryption in [CLLZ21].

- Let $\mathcal{P}_i = (P_i, I - P_i)$ be the following POVM acting on a quantum decryptor:
  - Sample a uniform $b \leftarrow \{0,1\}$ and random coins (which will be used to generated ciphertexts $\mathsf{ct}_1, \cdots, \mathsf{ct}_k$).
  - For each $j \in \{1, \cdots, i-1\}$, compute $\mathsf{ct}_j \leftarrow \mathsf{UD.Sim}(n)$ where n is the length of m_0 and m_1.
  - For each $j \in \{i, \cdots, k\}$, compute $\mathsf{ct}_j \leftarrow \mathsf{UD.Enc}(\mathsf{pk}_j, m_b)$.
  - Let $\mathsf{ct} = (\mathsf{ct}_1, \cdots, \mathsf{ct}_k)$.
  - Run the quantum decryptor on input ct. Check whether the outcome is m_b. If so, output 1; otherwise, output 0.
- Let PI_i be the projective implementation of $\mathcal{P}_i$.

It is easy to see that when a quantum decryptor is in the subspace defined by $\mathsf{TI}_{1/2+\gamma}$, applying PI_0 on the state will always produce a real number $\beta \geq 1/2 + \gamma$. This is a simple observation

```
On input $u = u_1||u_2||\cdots||u_\ell$ (where each $u_i \in \mathbb{F}_2^n$):
    1. If for all $i \in [\ell]$, $R_i^{r_i}(u_i) = 1$:
           Output $m$
    2. Else:
           Output $\perp$
```

Figure 5: Program $P_{m,r}$

following Theorem 2.7: $\mathsf{TI}_{1/2+\gamma}$ is implemented by first applying PI_0 and comparing the outcome with $1/2 + \gamma$.

Let the outcome of applying PI_0 on the i-th quantum decryptor of σ^* be a random variable $b_{i,0}$. We have:

$$\Pr\left[\forall i \in [k+1], b_{i,0} \geq \frac{1}{2} + \gamma\right] = 1. \tag{3}$$

Repeated Probability Measure and Its Properties. We then define repeated projective implementation. For the first quantum decryptor $\sigma^*[1]$, we apply PI_0 to obtain a outcome $b_{1,0}$. Then we apply the next projective implementation PI_1 on the leftover state to obtain a outcome $b_{1,1}$. So on and so forth, until we stop after applying PI_k. The outcomes of all measurements are denoted by random variables $b_{1,0}, \cdots, b_{1,k}$.

Claim 3.7. *There always exists $j \in [k]$ such that $b_{1,j-1} - b_{1,j} \geq \gamma/k$.*

Proof. For any quantum decryptor, if we apply PI_k on it, the outcome will always be $1/2$. This is because the ciphertext in PI_k is always generated without any information about m_0 or m_1. Therefore, every decryptor's behavior is random guessing: $b_{1,k}$ is always $1/2$.

From Equation (3), we know that $b_{1,0} \geq 1/2 + \gamma$. By triangle inequality, the claim holds. □

We use a random variable j_1 for the first index such that $b_{1,j_1-1} - b_{1,j_1} \geq \gamma/k$.

We similarly define the above repeated projective implementation for every quantum decryptor $\sigma^*[i]$. Since the repeated measurement on the i-th decryptor commutes with the repeated measurement on the i'-th ($i' \neq i$) decryptor, we can safely assume they are done in any order. Let $(b_{i,0}, \cdots, b_{i,j}, \cdots, b_{i,k})$ be the outcome of the repeated projective implementation the i-th decryptor. Similarly, Claim 3.7 holds for every decryptor:

Claim 3.8. *For every $i \in [k+1]$, there always exists $j \in [k]$ such that $b_{i,j-1} - b_{i,j} \geq \gamma/k$.*

Let j_i be the first index such that $b_{i,j_i-1} - b_{i,j_i} \geq \gamma/k$. We next show that there always exist $x \neq y$ such that $j_x = j_y$.

Claim 3.9. $\Pr[\exists x \neq y, j_x = j_y] = 1$.

Proof. This is simply because for every $i \in [k+1]$, $j_i \in [k]$. The claim follows from the pigeonhole principle. □

Guessing x, y and j_x. We describe the first half of our reduction algorithm. The algorithm takes as input σ^* (postselecting on all $\mathsf{TI}_{1/2+\gamma}$ output 1, and aborting if it fails). Below, we show the first part of the algorithm. In the second part of the reduction algorithm, it will extract a pair of secrets

On input the $k+1$ quantum decryptors σ^*:

1. Randomly sample $1 \leq x < y \leq k+1$ and $j \in [k]$;
2. Apply repeated projective measurement PI_0 to PI_{j-1} to $\sigma^*[x]$. Let $b_{x,j-1}$ be the last outcome.
3. Apply repeated projective measurement PI_0 to PI_{j-1} to $\sigma^*[y]$. Let $b_{y,j-1}$ be the last outcome.
4. Output $(x, y, j, b_{x,j-1}, b_{y,j-1})$ and both the x-th and y-th decryptors, denoted by $\sigma^{**}[x, y]$.

Figure 6: Reduction Algorithm Part 1

for the same coset states from $\sigma^{**}[x, y]$, which we will elaborate on shortly after.

We prove the following claim for the above algorithm.

Claim 3.10. *With probability at least $1/(2k^3)$, the above procedure produces $(x, y, j, b_{x,j-1}, b_{y,j-1})$ and $\sigma^{**}[x, y]$ satisfy:*

1. *Applying $\mathsf{PI}_{j-1}^{\otimes 2}$ jointly on $\sigma^*[x, y]$ produces $b_{x,j-1}, b_{y,j-1}$ with probability 1.*
2. *Applying $\mathsf{PI}_{j}^{\otimes 2}$ jointly on $\sigma^*[x, y]$ produces $b_{x,j}, b_{y,j}$, such that:*

$$\Pr\left[b_{x,j-1} - b_{x,j} \geq \frac{\gamma}{k} \wedge b_{y,j-1} - b_{y,j} \geq \frac{\gamma}{k}\right] \geq \frac{1}{2k^3}.$$

Proof for Claim 3.10. By Claim 3.8, there is always a pair of indices $x < y$ and an integer $j \in [k]$ such that $b_{x,j-1} - b_{x,j} \geq \frac{\gamma}{k}$ and $b_{y,j-1} - b_{y,j} \geq \frac{\gamma}{k}$ simultaneously. As a consequence, suppose that we guess x, y and j uniformly at random *after* applying the repeated projective implementation $\mathsf{PI}_0, \cdots, \mathsf{PI}_k$ on every quantum decryptor, then

$$\Pr\left[b_{x,j-1} - b_{x,j} \geq \frac{\gamma}{k} \wedge b_{y,j-1} - b_{y,j} \geq \frac{\gamma}{k}\right] \geq \frac{1}{\binom{k+1}{2} \cdot k} \geq \frac{1}{k^3}, \tag{4}$$

where the last inequality follows by $k \geq 1$.

Since the repeated projective implementations on disjoint quantum decryptors commute (with themselves as well as the final test projection), the same probability can be achieved if we *only* apply the repeated measurements on the x-th and y-th decryptors, skipping the other $(k-1)$ ones (see Figure 7).

We have

$$\Pr_{\mathsf{RandomMeasure}(\sigma^*)}\left[b_{x,j-1} - b_{x,j} \geq \frac{\gamma}{k} \wedge b_{y,j-1} - b_{y,j} \geq \frac{\gamma}{k}\right] \geq \frac{1}{k^3}. \tag{5}$$

Equation (4) and Equation (5) differ on how $b_{x,j-1}, b_{x,j}, b_{y,j-1}, b_{y,j}$ is sampled.

On input the $k+1$ quantum decryptors σ^*:

1. Randomly sample $1 \leq x < y \leq k+1$ and $j \in [k]$;
2. Apply repeated projective measurement PI_0 to PI_j to $\sigma^*[x]$. Let $b_{x,j-1}, b_{x,j}$ be the last two outcomes.
3. Apply repeated projective measurement PI_0 to PI_j to $\sigma^*[y]$. Let $b_{y,j-1}, b_{y,j}$ be the last two outcomes.
4. Output $(x, y, j, b_{x,j-1}, b_{x,j}, b_{y,j-1}, b_{y,j})$.

Figure 7: Algorithm $\mathsf{RandomMeasure}(\sigma^*)$

We can view our reduction algorithm (Figure 6) as the first step of $\mathsf{RandomMeasure}$ (Figure 7). More formally, $\mathsf{RandomMeasure}$ first runs the reduction algorithm to get $(x, y, j, b_{x,j-1}, b_{y,j-1})$ and $\sigma^{**}[x, y]$; it then applies PI_j on both registers to obtain $b_{x,j}$ and $b_{y,j}$.

If the claim we want to prove does not hold, then with probability $< 1/(2k^3)$, the outcome $(x, y, j, b_{x,j-1}, b_{y,j-1})$ and $\sigma^{**}[x, y]$ satisfy condition (2) in Claim 3.10. Therefore, the probability in Equation (5) is strictly smaller than $1/(2k^3) + 1/(2k^3)$. This is a contradiction, as again the test projector commutes with the rest of $\mathsf{RandomMeasure}$. □

Extracting Secrets from $\sigma^{**}[x, y]$. We describe the second half of our reduction algorithm. Given $(x, y, j, b_{x,j-1}, b_{y,j-1})$ and $\sigma^{**}[x, y]$ that satisfy both conditions in Claim 3.10, we can extract secrets for both coset states. This violates the strong computational monogamy-of-entanglement property of coset states, thus finishes the proof.

Recall the underlying ciphertext distribution of PI_{j-1} and PI_j:

1. The first $j-1$ ciphertexts $\mathsf{ct}_1, \cdots, \mathsf{ct}_{j-1}$ are generated by $\mathsf{Sim}(n)$.
2. The last $k-j$ ciphertexts $\mathsf{ct}_{j+1}, \cdots, \mathsf{ct}_k$ are generated honestly, using their corresponding public key.
3. The j-th ciphertext is either generated honestly using the j-th public key pk_j (in PI_{j-1}), or by $\mathsf{Sim}(n)$ (in PI_j). $\mathsf{pk}_j, \mathsf{sk}_j$ is generated by $\mathsf{UD.Setup}$ in Figure 4. Let the underlying cosets be $\{A_l + s_l, A_l^\perp + s'_l\}_{l=1}^{\ell}$:

$$\mathsf{pk}_j = \{\mathsf{iO}(A_l + s_l), \mathsf{iO}(A_l^\perp + s'_l)\}_{l \in [\ell]},$$
$$\mathsf{sk}_j = \{A_l, s_l, s'_l\}_{l \in [\ell]}.$$

The following claim says that if applying P_{j-1} or P_j on a quantum decryptor produce different values (with difference more than γ/k), then we can extract ℓ vectors $v_1, \cdots, v_\ell$: each v_l is uniformly in either $A_l + s_l$ or $A_l^\perp + s'_l$.

Claim 3.11. *For any $k = \mathsf{poly}(\lambda)$, let $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{CRUD.Setup}(1^\lambda, k)$ where $\mathsf{sk} = (\mathsf{sk}_1, \cdots, \mathsf{sk}_k)$ and $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$. Let ρ_{sk} be the unclonable decryption key. For any $j \in [k]$, let PI_{j-1} and PI_j be defined at the beginning of the proof. Let $\mathsf{pk}_j = \{\mathsf{iO}(A_l + s_l), \mathsf{iO}(A_l^\perp + s'_l)\}_{l \in [\ell]}, \mathsf{sk}_j = \{A_l, s_l, s'_l\}_{l \in [\ell]}$.*

If there exist inverse polynomials $\alpha_1(\cdot), \alpha_2(\cdot)$ and an quantum algorithm $\mathcal{B}$ that takes $(\rho_{\mathsf{sk}}, \mathsf{pk})$ outputs ρ such that with probability at least α_1, ρ satisfies the following:

1. *There exists $b_{j-1} \in (0, 1]$, applying PI_{j-1} on ρ always produces b_{j-1}.*

2. *Let the outcome of applying* PI_j *on* ρ *be* b_j*. Then* $\Pr[b_{j-1} - b_j > \gamma/k] > \alpha_2$*.*

Then there exists another inverse polynomial $\beta(\cdot)$ *and an efficient quantum algorithm* $\mathcal{C}$ *that takes all the descriptions of* $\{A_l\}_{l=1}^{\ell}$ *(denoted by* $\mathbf{A}$*),* ρ *and* ℓ *random coins* $r_1, \cdots, r_\ell \in \{0,1\}$ *such that:*

$$\Pr_{\substack{\mathsf{sk},\mathsf{pk},\rho_{\mathsf{sk}},r \\ \rho \leftarrow \mathcal{B}(\rho_{\mathsf{sk}},\mathsf{pk})}} \left[\forall l \in [\ell], v_l \in \begin{cases} A_l + s_l & \text{if } r_l = 0 \\ A_l^\perp + s'_l & \text{if } r_l = 1 \end{cases}, (v_1, \cdots, v_\ell) \leftarrow \mathcal{C}(\mathbf{A}, \rho, r) \right] \geq \beta.$$

The proof of this is similar to the extraction technique in [CLLZ21] using compute-and-compare obfuscation. We refer interested readers to Appendix B.

By setting $\alpha_1 = \alpha_2 := 1/(2k^3)$, $\mathcal{B}$ be the reduction algorithm in Figure 6 and $\rho := \sigma^{**}[x]$, we conclude that there exists another algorithm that takes $\sigma^{**}[x]$, random coins $r_1, \cdots, r_\ell$ and outputs $(v_1, \cdots, v_\ell)$ in the corresponding cosets (depending on each r_l).

Next, we show that after a successful extraction on the $\sigma^{**}[x]$, the other decryptor still satisfy the conditions (1) (2) for Claim 3.11. Therefore, we can extract another random set of vectors from the other decryptor, with non-negligible probability, even conditioned on a successful extraction on $\sigma^{**}[x]$.

Assume conditioned on a successful extraction on the $\sigma^{**}[x]$, the other decryptor becomes $\sigma'[y]$ and it does not satisfy the conditions in Claim 3.11.

First, applying PI_{j-1} on $\sigma'[y]$ always produces $b_{y,j-1}$. This is because the extraction on the $\sigma^{**}[x]$ register does not change the support of $\sigma'[y]$. Thus, condition (2) in Claim 3.11 can not hold. Let $\mathbf{E}_1$ denote a successful (E)xtraction on $\sigma^{**}[x]$ and $\mathbf{G}_2$ be a indicator that applying PI_j on $\sigma^{**}[y]$ to get $b_{y,j}$ and $b_{y,j} < b_{y,j-1} - \frac{\gamma}{k^3}$ (a big (G)ap). We know that in this case, $\Pr[\mathbf{E}_1 \wedge \mathbf{G}_2]$ is negligibly small.

However, this can not be true. We can imagine PI_j is implemented first. We know that $\Pr[\mathbf{G}_2]$ is non-negligible by the condition (2) in Claim 3.10. Conditioned on $\mathbf{G}_2$, let the x-th decryptor become $\sigma'[x]$. We know that $\sigma'[x]$ must satisfy both conditions in Claim 3.11. Otherwise, condition (2) in Claim 3.10 can not hold. Thus, $\Pr[\mathbf{G}_2|\mathbf{E}_1]$ must be non-negligible. This contradicts with the assumption that $\Pr[\mathbf{E}_1 \wedge \mathbf{G}_2]$ is negligibly small.

Thus, the reduction algorithm, with non-negligible probability, can extract $(v_1, \cdots, v_\ell)$ and $(v'_1, \cdots, v'_\ell)$ with respect to random $r_1, \cdots, r_\ell$ and $r'_1, \cdots, r'_\ell$. With probability at least $1 - 2^{-\ell}$, there exist $l \in [\ell]$ such that $r_l \neq r'_l$. Thus, v_l and v'_l will be two vectors in each of the cosets $A_l + s_l$ and $A'_l + s'_l$. By guessing this l, this breaks the computational strong monogamy-of-entanglement game (Theorem 2.5). □

4 Collusion Resistant Copy-Protection for Signature Schemes

In this section, we present security definiton and the construction for copy-protecting signatures and PRFs.

4.1 Copy-Protection for Signatures: Definitions

Definition 4.1 (Bounded Collusion Resistant Copy-Protection Scheme of Signature Scheme)**.** *A bounded collusion resistant copy-protection Scheme for a signature scheme consists of the following algorithms:*

$\mathsf{Setup}(1^\lambda, k)$*: takes in security parameter* 1^λ *and upper bound* k*; outputs classical secret key* sk *and classical verification key* vk*;*

$\mathsf{QKeyGen}(\mathsf{sk})$*: takes in a classical secret key* sk*; outputs* k *quantum signing keys* $\rho_{\mathsf{sk}} = \rho_{\mathsf{sk},1} \otimes \rho_{\mathsf{sk},2} \otimes \cdots \otimes \rho_{\mathsf{sk},k}$

$\mathsf{Sign}(\rho_{\mathsf{sk}}, x)$*: takes a quantum signing key* ρ_K *and an input* $x \in [N]$*; outputs a classical signature* $\mathsf{sig} \in [M]$.

$\mathsf{Verify}(\mathsf{vk}, x, \mathsf{sig})$*: takes in verification key* vk*, message* x *and claimed signature* sig*. It outputs 1 (accept) or 0 (reject).*

A copy-protection for signatures scheme should satisfy the following properties:

Correctness For every polynomial $k(\cdot)$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all λ, all messages x, all $i \in [k]$:

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, x, \mathsf{sig}) = 1 \;\middle|\; \begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, k), \\ \rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}), \\ \mathsf{sig} \leftarrow \mathsf{Sign}(\rho_{\mathsf{sk},i}, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Multi-time Correctness/Reusability To ensure that the quantum signing key is reusable (i.e., satisfying the above correctness condition) for polynomially many times, we require the following property for our scheme:

- Pseudo-deterministic Signing Procedure: For all polynomial $k(\cdot)$, there is a negligible function $\mathsf{negl}(\cdot)$ for all λ, for all messages x, for all $(\mathsf{sk}, \mathsf{vk})$ in the support of $\mathsf{Setup}(1^\lambda, k)$ and all $i \in [k]$, there exists a signature sig^* such that the following holds:

$$\Pr\left[\mathsf{sig} = \mathsf{sig}^* \;\middle|\; \begin{array}{c} \rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}), \\ \mathsf{sig} \leftarrow \mathsf{Sign}(\rho_{\mathsf{sk},i}, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Therefore, the quantum signing key can be used for polynomially many times, by the gentle measurement lemma [Aar05].

Remark 4.2. *The construction for the signature scheme in this work consists of a deterministic signing procedure and hence satisfies reusability.*

An alternative requirement for reusability is to allow non-deterministic signatures, but require correctness for any polynomial length sequence of messages. We will not elaborate on this direction.

Existential Unforgeability This is the standard (selective) existential unforgeability under chosen message attack game for signature schemes. The adversary is *not* given any copy of the signing key, but only oracle access to the signing function.

1. The adversary gives the challenger the message x^*.
2. The challenger samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, k)$. It gives vk to $\mathcal{A}$;
3. The adversary queries the signing oracle for a polynomial number of times on messages $x \neq x^*$.

4. The adversary provides a signature sig^* for message x^*. The challenger accepts if and only if $\mathsf{Verify}(\mathsf{vk}, x^*, \mathsf{sig}^*) = 1$.

For any QPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all λ, x, the probability that $\mathcal{A}$ wins the above game is $\mathsf{negl}(\lambda)$.

Remark 4.3. *In the above game, we only allow the adversary to query the signing oracle classically. Classical query access is a more reasonable assumption in the security game for signatures since the signing oracle is in the hands of the challenger, who can choose to interact with the adversary through a classical channel (unlike other settings, for example in case of a random oracle, the hash function modeled as the oracle is in the hands of the adversary and can then be queried in superposition).*

Anti-Piracy Security for k-bounded collusion resistance Let $\lambda \in \mathbb{N}^+$. Consider the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, k)$ and $\rho_{\mathsf{sk}} = \rho_{\mathsf{sk},1} \otimes \rho_{\mathsf{sk},2} \otimes \cdots \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$. It gives ρ_{sk} and vk to $\mathcal{A}$;
2. $\mathcal{A}$ returns to the challenger a (possibly mixed and entangled) state σ on registers R_1, R_2, $\cdots, R_{k+1}$. We interpret σ as $k+1$ (possibly entangled) quantum programs $\sigma[R_1], \cdots, \sigma[R_{k+1}]$.
3. The challenger samples uniformly random $x_1, \cdots, x_{k+1} \leftarrow [N]$. Then runs a universal circuit on input $(\sigma[R_i], x_i)$ to obtain sig_i' for each $i \in [k+1]$. The outcome of the game is 1 if and only if $\mathsf{Verify}(\mathsf{vk}, x_i, \mathsf{sig}_i') = 1$ for all $i \in [k+1]$.

Denote by $\mathsf{CPSignatureGame}(1^\lambda, \mathcal{A})$ a random variable for the output of the game. We say the scheme has anti-piracy security if for every polynomial-time quantum algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $\lambda \in \mathbb{N}^+$,

$$\Pr\left[\mathsf{CPSignatureGame}(1^\lambda, \mathcal{A}) = 1\right] = \mathsf{negl}(\lambda).$$

4.2 Construction

In this section, we describe a construction of a copy-protection scheme for signatures. Our construction is based on the copy-protection scheme for PRFs in [CLLZ21] and the short signature scheme in [SW21].

Let λ be the security parameter. Our copy-protection construction for a signature will make use of the following building blocks. We refer the readers to Appendix A.3 for details on the PRF building blocks we use.

1. A puncturable $F_1 : [K_\lambda] \times [N_\lambda] \to [M_\lambda]$, where $N = 2^{n(\lambda)}$ and $M = 2^{m(\lambda)}$, for some polynomials $n(\lambda)$ and $m(\lambda)$, satisfying $n(\lambda) \geq m(\lambda) + 2\lambda + 4$. For convenience, we will omit writing the dependence on λ, when it is clear from the context.
   F_1 is also an extracting PRF with error $2^{-\lambda-1}$ for min-entropy $k(\lambda) = n(\lambda)$ (i.e., a uniform distribution over all possible inputs). By Theorem A.14, such PRFs exist assuming post-quantum one-way functions.
2. A puncturable statistically injective PRF F_2 with failure probability $2^{-\lambda}$ that accepts inputs of length ℓ_2 and outputs strings of length ℓ_1. By Theorem A.13, such a PRF exists assuming one-way functions exist, and as long as $\ell_1 \geq 2\ell_2 + \lambda$.

3. A puncturable PRF F_3 that accepts inputs of length ℓ_1 and outputs strings of length ℓ_2. By Lemma C.4 in [SW21], assuming one-way functions exist, F_3 is a puncturable PRF.
4. A one-way function $\mathsf{OWF} : [M_\lambda] \to [M_\lambda]$.

In our construction, we will parse the input x to PRF $F_1(K_1, \cdot)$ as three substrings $x_0||x_1||x_2$, where each x_i is of length ℓ_i for $i \in \{0, 1, 2\}$ and $n = \ell_0 + \ell_1 + \ell_2$. $\ell_2 - \ell_0$ should also be large enough (we will specify later how large).

Next, we describe a copy-protection scheme for a signing key, using the above building blocks. The description is contained in Figures 8, 9 and 10.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$:

- Sample PRF keys K_1, K_2, K_3 for F_1, F_2, F_3.
- Let VK be the program in Figure 10. Prepare obfuscated program $\mathsf{iO}(\mathsf{VK})$.
- Output $\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk} = \mathsf{iO}(\mathsf{VK})$

$\mathsf{QKeyGen}(\mathsf{sk}) \to \rho_{\mathsf{sk}}$:

- Sample $\{A_i, s_i, s'_i\}_{i\in[\ell]}$: uniformly random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s'_i for $i = 1, 2, \cdots, \ell_0$.
- Let CPSign be the program described in Figure 9. Prepare $\mathsf{iO}(\mathsf{CPSign})$.
- Output the quantum key $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$,

$\mathsf{Sign}(\rho_{\mathsf{sk}}, x) \to \mathsf{sig}$:

- Let $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$. Parse x as $x = x_0||x_1||x_2$ where x_0 is of length ℓ_0.
- For all $i \in [\ell_0]$, if $x_{0,i}$ is 1, apply $H^{\otimes n}$ to $|A_{i,s_i,s'_i}\rangle$. Otherwise, leave the state unchanged.
- We obtain state σ from the above procedure (which can be seen as a superposition over tuples of l_0 vectors). Run $\mathsf{iO}(\mathsf{CPSign})$ coherently on input x and σ, and measure the final output register to obtain sig.

$\mathsf{Verify}(\mathsf{vk}, x, \mathsf{sig}) \to 0/1$:

- Parse vk as program $\mathsf{iO}(\mathsf{VK})$.
- Run $\mathsf{iO}(\mathsf{VK})$ on input (x, sig) to obtain output $0/1$.

Figure 8: Quantum copy-protection scheme for digital signature.

Signing Program The program CPSign, described in Figure 9, takes as input x and ℓ_0 vectors $v_1, \cdots, v_{\ell_0}$, and has two modes. If x is not in the sparse hidden trigger set (not passing the 'if' check in the first line), the program is in the *normal mode*: it outputs the PRF evaluation of $F_1(K_1, x)$ if and only if every v_i is in the appropriate coset. Otherwise, the program is in the *hidden trigger mode*: in this mode, a classical circuit description Q' is computed from the input x; the program then outputs $Q'(v_1, \cdots, v_{\ell_0})$.

On almost all inputs except a sparse set of hidden triggers, the program runs in its normal mode. For $i \in [l_0]$, define the programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$ (where the inputs to iO should be appropriately padded).

Note that in the normal mode, the signature for a message x output by the program CPSign is the PRF evaluation $F_1(K_1, x)$.

```
Hardcoded: Keys $K_1, K_2, K_3$, $R_i^0, R_i^1$ for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$:
  1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
       Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output
       $Q'(\mathsf{mode} = \mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
  2. Otherwise, check if the following holds: for all $i \in [\ell_0]$, $R_i^{x_{0,i}}(v_i) = 1$ (where $x_{0,i}$ is the $i$-th bit of $x_0$).
       Normal Mode: If so, output $F_1(K_1, x)$. Otherwise, output $\perp$.
```

Figure 9: Program CPSign

```
Hardcoded: Keys $K_1, K_2, K_3$.
On input $x = x_0||x_1||x_2$ and $\mathsf{sig}$:
  1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
       Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output $Q'(\mathsf{mode} = \mathsf{check}, \mathsf{sig}||0^{\ell_0 \cdot \lambda - m})$, where sig is padded with 0's to the length of $\ell_0 \cdot \lambda$.
  2. Otherwise:
       Normal Mode: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Otherwise, output 0.
```

Figure 10: Program VK

Verification Program The program used for verification of signatures, VK, also has two modes similar to CPSign. VK's differences from CPSign are highlighted in blue. It takes a message x and a claimed signature sig as inputs.

In the Hidden Trigger Mode, we compute a circuit Q' from the input x same as we do in CPSign, but we input a mode indicator check into circuit Q' as well as the claimed signature sig padded to the same length as $v_1, \cdots, v_{\ell_0}$. In the normal mode, we check if the one-way function evaluation $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. Both modes will eventually output a 0 or 1.

Note that the circuit Q' used in the hidden trigger mode also has two modes(eval mode and check mode) inside itself. We will specify what they are in the security proof (Appendix C.1 and Figure 12).

Theorem 4.4. *The construction in Figure 8 satisfies selective existential unforgeability and anti-piracy security for 1-bounded collusion resistance as defined in Section 4.1.*

We give a proof for the above theorem in Appendix C.

4.3 Collusion Resistant Copy-Protection for Signatures and PRFs

The k-bounded collusion resistant copy-protection for signatures can be generalized naturally from the above 1-bounded collusion resistant one. However, the construction is not completely a black-box one that simply runs the single-copy scheme for k times. We will elaborate the algorithms and proofs in Appendix E.

The collusion resistant copy-protection for PRFs bears a lot of similarity with the signature construction. We will describe the construction and proofs in Appendix G.

References

[Aar05] Scott Aaronson. "Limitations of Quantum Advice and One-Way Communication". In: *Theory of Computing* 1.1 (2005), pp. 1–28. DOI: `10.4086/toc.2005.v001a001` (cit. on pp. 17, 26, 57).

[Aar09] Scott Aaronson. "Quantum Copy-Protection and Quantum Money". In: *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*. 2009, pp. 229–242. DOI: `10.1109/CCC.2009.42` (cit. on pp. 1, 2, 4, 13).

[AC13] Scott Aaronson and Paul Christiano. "Quantum Money from Hidden Subspaces". In: *Theory of Computing* 9.9 (2013), pp. 349–401. DOI: `10.4086/toc.2013.v009a009` (cit. on p. 3).

[AK22] Prabhanjan Ananth and Fatih Kaleoglu. *A Note on Copy-Protection from Random Oracles*. 2022. DOI: `10.48550/ARXIV.2208.12884`. URL: `https://arxiv.org/abs/2208.12884` (cit. on p. 5).

[AKL$^+$22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. "On the Feasibility of Unclonable Encryption, and More". In: *Advances in Cryptology - CRYPTO 2022 - 42st Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings*. Vol. 13507. 2022 (cit. on pp. 2, 4).

[ALL$^+$21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. "New Approaches for Quantum Copy-Protection". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Vol. 12825. 2021, pp. 526–555. DOI: `10.1007/978-3-030-84242-0_19` (cit. on pp. 2–6, 12, 15, 16, 35).

[AP21] Prabhanjan Ananth and Rolando L. La Placa. "Secure Software Leasing". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. Vol. 12697. 2021, pp. 501–530. DOI: `10.1007/978-3-030-77886-6_17` (cit. on pp. 2, 3, 5).

[BB84] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Proceedings of International Conference on Computers, Systems & Signal Processing, Dec. 9-12, 1984, Bangalore, India*. 1984, pp. 175–179. arXiv: `2003.06557` (cit. on p. 1).

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. Vol. 2139. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1 (cit. on pp. 3, 14).

[BJL$^+$21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. "Secure Software Leasing Without Assumptions". In: *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*. Vol. 13042. 2021, pp. 90–120. DOI: 10.1007/978-3-030-90459-3_4 (cit. on pp. 2, 3, 5).

[BS16] Shalev Ben-David and Or Sattath. *Quantum Tokens for Digital Signatures*. 2016. arXiv: 1609.09047 (cit. on p. 12).

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. "Tight quantum time-space tradeoffs for function inversion". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 673–684 (cit. on p. 38).

[CHN$^+$18] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. "Watermarking cryptographic capabilities". In: *SIAM Journal on Computing* 47.6 (2018), pp. 2157–2202 (cit. on p. 3).

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. "Hidden Cosets and Applications to Unclonable Cryptography". In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Vol. 12825. 2021, pp. 556–584. DOI: 10.1007/978-3-030-84242-0_20 (cit. on pp. 2–4, 6–8, 10, 12, 14, 15, 18, 19, 21, 25, 27, 33, 34, 41, 43–45, 48, 58, 59).

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. *Quantum copy-protection of compute-and-compare programs in the quantum random oracle model*. 2020. URL: https://arxiv.org/abs/2009.13865 (cit. on pp. 2–5).

[CV21] Eric Culf and Thomas Vidick. *A monogamy-of-entanglement game for subspace coset states*. 2021. arXiv: 2107.13324 (cit. on p. 15).

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits". In: *SIAM Journal on Computing* 45.3 (2016), pp. 882–929. DOI: 10.1137/14095772X (cit. on p. 14).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions". In: *J. ACM* 33.4 (1986), pp. 792–807. DOI: 10.1145/6490.6503 (cit. on p. 36).

[GKM$^+$19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. "Watermarking Public-Key Cryptographic Primitives". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*. Vol. 11694. 2019, pp. 367–398. DOI: 10.1007/978-3-030-26954-8_12 (cit. on pp. 3, 4).

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. "Lockable Obfuscation". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 612–621. DOI: 10.1109/FOCS.2017.62 (cit. on p. 34).

[GKWW21] Rishab Goyal, Sam Kim, Brent Waters, and David J. Wu. "Beyond Software Watermarking: Traitor-Tracing for Pseudorandom Functions". In: *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*. Vol. 13092. 2021, pp. 250–280. DOI: 10.1007/978-3-030-92078-4_9 (cit. on p. 58).

[GZ20] Marios Georgiou and Mark Zhandry. *Unclonable Decryption Keys*. 2020. Cryptology ePrint Archive: 2020/877. URL: https://eprint.iacr.org/2020/877 (cit. on pp. 4, 12).

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. "Secure Software Leasing from Standard Assumptions". In: *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*. Vol. 13042. 2021, pp. 31–61. DOI: 10.1007/978-3-030-90459-3_2 (cit. on pp. 3, 5).

[Kre21] William Kretschmer. "Quantum Pseudorandomness and Classical Complexity". In: *16th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2021, July 5-8, 2021, Virtual Conference*. Vol. 197. 2021, 2:1–2:20. DOI: 10.4230/LIPIcs.TQC.2021.2 (cit. on p. 2).

[KW17] Sam Kim and David J Wu. "Watermarking cryptographic functionalities from standard lattice assumptions". In: *Annual International Cryptology Conference*. Springer. 2017, pp. 503–536 (cit. on p. 3).

[KW19] Sam Kim and David J Wu. "Watermarking PRFs from lattices: stronger security via extractable PRFs". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 335–366 (cit. on p. 3).

[MW05] Chris Marriott and John Watrous. "Quantum Arthur–Merlin games". In: *computational complexity* 14.2 (2005), pp. 122–152. DOI: 10.1007/s00037-005-0194-x (cit. on pp. 6, 34).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667 (cit. on p. 14).

[SW21] Amit Sahai and Brent Waters. "How to Use Indistinguishability Obfuscation: Deniable Encryption, and More". In: *SIAM Journal on Computing* 50.3 (2021), pp. 857–908. DOI: 10.1137/15M1030108 (cit. on pp. 12, 14, 27, 28, 36, 39, 41, 52).

[VZ21] Thomas Vidick and Tina Zhang. "Classical Proofs of Quantum Knowledge". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. Vol. 12697. 2021, pp. 630–660. DOI: 10.1007/978-3-030-77886-6_22 (cit. on p. 14).

[Wie83] Stephen Wiesner. "Conjugate coding". In: *SIGACT News* 15.1 (1983), pp. 78–88. DOI: 10.1145/1008908.1008920 (cit. on p. 1).

[WZ17] Daniel Wichs and Giorgos Zirdelis. "Obfuscating Compute-and-Compare Programs under LWE". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 600–611. DOI: `10.1109/FOCS.2017.61` (cit. on p. 34).

[YAL+19] Rupeng Yang, Man Ho Au, Junzuo Lai, Qiuliang Xu, and Zuoxia Yu. "Collusion Resistant Watermarking Schemes for Cryptographic Functionalities". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*. Vol. 11921. 2019, pp. 371–398. DOI: `10.1007/978-3-030-34578-5_14` (cit. on pp. 3, 4).

[YAYX20] Rupeng Yang, Man Ho Au, Zuoxia Yu, and Qiuliang Xu. "Collusion Resistant Watermarkable PRFs from Standard Assumptions". In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*. Vol. 12170. 2020, pp. 590–620. DOI: `10.1007/978-3-030-56784-2_20` (cit. on p. 3).

[Zha19] Mark Zhandry. "The Magic of ELFs". In: *Journal of Cryptology* 32.3 (2019), pp. 825–866. DOI: `10.1007/s00145-018-9289-9` (cit. on p. 34).

[Zha20] Mark Zhandry. "Schrödinger's Pirate: How to Trace a Quantum Decoder". In: *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*. Vol. 12552. 2020, pp. 61–91. DOI: `10.1007/978-3-030-64381-2_3` (cit. on pp. 6, 16, 34, 35).

[Zha21] Mark Zhandry. "How to Construct Quantum Random Functions". In: *J. ACM* 68.5 (2021). DOI: `10.1145/3450745` (cit. on p. 36).

A Additonal Prelimanaries

A.1 Compute-and-Compare Obfuscation with Quantum Auxiliary Input

In this section, we recall the definition of compute-and-compare obfuscation with quantum auxiliary input for unpredictable distributions, first discussed in [CLLZ21].

Definition A.1 (Compute-and-Compare Program). *Given a function $f : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}$ along with a target value $y \in \{0,1\}^{\ell_{\mathsf{out}}}$ and a message $z \in \{0,1\}^{\ell_{\mathsf{msg}}}$, we define the compute-and-compare program:*

$$\mathsf{CC}[f,y,z](x) = \begin{cases} z & \text{if } f(x) = y \\ \perp & \text{otherwise} \end{cases}.$$

We define the following class of *unpredictable distributions* over pairs of the form $(\mathsf{CC}[f,y,z], \mathsf{aux})$, where aux is auxiliary quantum information. These distributions are such that y is computationally unpredictable given f, z and aux.

Definition A.2 (Unpredictable Distributions). *We say that a family of distributions $D = \{D_\lambda\}$ where D_λ is a distribution over pairs of the form $(\mathsf{CC}[f,y,z], \mathsf{aux})$ where aux is a quantum state, belongs to the* *class of* unpredictable distributions *if the following holds. There exists a negligible function* negl *such that, for all QPT algorithms* $\mathcal{A}$*,*

$$\Pr_{(\mathsf{CC}[f,y,z],\mathsf{aux})\leftarrow D_\lambda}\left[A(1^\lambda, f, z, \mathsf{aux}) = y\right] \leq \mathsf{negl}(\lambda).$$

We further define the class of *sub-exponentially unpredictable distributions*, where we require the guessing probability to be inverse sub-exponential in the security parameter.

Definition A.3 (Sub-Exponentially Unpredictable Distributions)**.** *We say that a family of distributions $D = \{D_\lambda\}$ where D_λ is a distribution over pairs of the form $(\mathsf{CC}[f, y, z], \mathsf{aux})$ where aux is a quantum state, belongs to the class of* sub-exponentially unpredictable distributions *if the following holds. There exists a sub-exponential function* subexp *such that, for all QPT algorithms* $\mathcal{A}$*,*

$$\Pr_{(\mathsf{CC}[f,y,z],\mathsf{aux})\leftarrow D_\lambda}\left[A(1^\lambda, f, z, \mathsf{aux}) = y\right] \leq 1/\mathsf{subexp}(\lambda).$$

Each program P has an associated set of parameters $P.\mathsf{param}$ (e.g input size, output size, circuit size), which is revealed to everyone.

Definition A.4 (Compute-and-Compare Obfuscation)**.** *A PPT algorithm* $\mathsf{CC.Obf}$ *is an obfuscator for the class of unpredictable distributions (or sub-exponentially unpredictable distributions) if for any family of distributions $D = \{D_\lambda\}$ belonging to the class, the following holds:*

- *Functionality Preserving: there exists a negligible function* negl *such that for all λ, every program P in D_λ,*

$$\Pr[\forall x,\ \widetilde{P}(x) = P(x),\ \widetilde{P} \leftarrow \mathsf{CC.Obf}(1^\lambda, P)] \geq 1 - \mathsf{negl}(\lambda)$$

- *Distributional Virtual-Black-Box: there exists an efficient simulator* Sim *such that:*

$$(\mathsf{CC.Obf}(1^\lambda, P), \mathsf{aux}) \approx_c (\mathsf{Sim}(1^\lambda, P.\mathsf{param}), \mathsf{aux})$$

  where $(P, \mathsf{aux}) \leftarrow D_\lambda$.

Combining the results of [WZ17, GKW17] with those of [Zha19], Coladangelo et al. [CLLZ21] showed the following theorem.

Theorem A.5. *Assuming the existence of post-quantum* iO *and the sub-exponential quantum hardness of LWE, there exist obfuscators for sub-exponentially unpredictable distributions, as in Definition A.4.*

A.2 Measure Success Probabilities of Quantum Adversaries, Efficiently

Approximating Threshold Implementation *Projective* and *threshold* implementations of POVMs are unfortunately not efficiently computable in general. Fortunately, they can be approximated, as shown by Zhandry [Zha20], using a technique first introduced by Marriott and Watrous [MW05].

We start with an efficient version of PI:

Theorem A.6 (Approximated Projective Implementation, Theorem 6.2 in [Zha20])**.** *Let $\mathcal{D}$ be a distribution of inputs. Let $\mathcal{P}_\mathcal{D} = (P_\mathcal{D}, Q_\mathcal{D})$ be a binary outcome POVM described above with respect to the distribution $\mathcal{D}$. For any $\delta, \epsilon > 0$, there exists an efficient procedure $\mathsf{API}^{\epsilon,\delta}(\mathcal{P}_\mathcal{D})$ such that:*

(i) If $\mathsf{API}^{\epsilon,\delta}(\mathcal{P}_{\mathcal{D}})$ on some quantum program outputs (ρ, p), applying $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ on ρ yields (ρ', p') satisfying $p' \geq p - \epsilon$ with probability at least $1 - \delta$.

(ii) If $\mathsf{PI}(\mathcal{P}_{\mathcal{D}})$ on some quantum program outputs (ρ, p), applying $\mathsf{API}^{\epsilon,\delta}(\mathcal{P}_{\mathcal{D}})$ on ρ yields (ρ', p') satisfying $p' \geq p - \epsilon$ with probability at least $1 - \delta$.

(iii) The running time of $\mathsf{API}^{\epsilon,\delta}(\mathcal{P}_{\mathcal{D}})$ is polynomial in the time complexity of sampling $\mathcal{D}$, the time complexity of running ρ, $1/\epsilon$ and $\log(1/\delta)$.

When the distribution is clear from the context, we sometimes ignore the subscript $\mathcal{D}$ in $\mathsf{API}^{\epsilon,\delta}(\mathcal{P}_{\mathcal{D}})$.

We will make use of the following lemma from a subsequent work of Aaronson et al. [ALL+21].

Theorem A.7 (Approximated Threshold Implementation, Corollary 1 in [ALL+21]). *Let $\mathcal{D}$ be a distribution of inputs. Let $\mathcal{P}_{\mathcal{D}} = (P_{\mathcal{D}}, Q_{\mathcal{D}})$ be a binary outcome POVM described above with respect to the distribution $\mathcal{D}$. For any $\delta, \epsilon > 0$ and γ, there exists an efficient procedure $\mathsf{ATI}^{\epsilon,\delta}_{\gamma}(\mathcal{P}_{\mathcal{D}})$ such that:*

(i) If $\mathsf{ATI}^{\epsilon,\delta}_{\gamma}(\mathcal{P}_{\mathcal{D}})$ on some quantum program outputs $(\rho, b = 1)$, applying $\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_{\mathcal{D}})$ on ρ yields (ρ', b') satisfying $b' = 1$ with probability at least $1 - \delta$.

(ii) If $\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_{\mathcal{D}})$ on some quantum program outputs $(\rho, b = 1)$, applying $\mathsf{ATI}^{\epsilon,\delta}_{\gamma}(\mathcal{P}_{\mathcal{D}})$ on ρ yields (ρ', b') satisfying $b' = 1$ with probability at least $1 - \delta$.

(iii) The running time of $\mathsf{ATI}^{\epsilon,\delta}_{\gamma}(\mathcal{P}_{\mathcal{D}})$ is polynomial in the time complexity of sampling $\mathcal{D}$, the time complexity of running ρ, $1/\epsilon$ and $\log(1/\delta)$ (independent of γ).

When the distribution is clear from the context, we sometimes ignore the subscript $\mathcal{D}$ in $\mathsf{ATI}^{\epsilon,\delta}_{\gamma}(\mathcal{P}_{\mathcal{D}})$.

The following lemma will be important in our proofs. Let D_0 and D_1 be two computationally indistinguishable distributions. Let $\gamma, \gamma' > 0$ be inverse-polynomially close. Then for any (efficiently constructible) state ρ, the probabilities of obtaining outcome 1 upon measuring $\mathsf{TI}_{\gamma}(\mathcal{P}_{D_0})$ and $\mathsf{TI}_{\gamma'}(\mathcal{P}_{D_1})$ respectively are negligibly close.

Theorem A.8 (Theorem 6.5 in [Zha20]). *Let $\gamma > 0$. Let ρ be an efficiently constructible mixed state, and let D_0, D_1 be two efficiently sampleable and computationally indistinguishable distributions. For any inverse polynomial ϵ, there exists a negligible function δ such that*

$$\mathrm{Tr}[\mathsf{TI}_{\gamma-\epsilon}(\mathcal{P}_{D_1})\rho] \geq \mathrm{Tr}[\mathsf{TI}_{\gamma}(\mathcal{P}_{D_0})\rho] - \delta\,.$$

A.3 Preliminaries: Puncturable PRFs and related notions

A *puncturable* PRF is a PRF equipped with an additional algorithm that "punctures" a PRF key K at a set of points S, so that the adversary with the punctured key can evaluate the PRF at all points except the points in S. Even given the punctured key, an adversary cannot distinguish between a uniformly random value and the evaluation of the PRF at a point S using the original unpunctured key. Formally:

Definition A.9 ((Post-quantum) Puncturable PRF). *A PRF family $F : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ with key generation procedure KeyGen_F is said to be puncturable if there exists an algorithm $\mathsf{Puncture}_F$, satisfying the following conditions:*

- ***Functionality preserved under puncturing:*** *Let $S \subseteq \{0,1\}^{n(\lambda)}$. For all $x \in \{0,1\}^{n(\lambda)}$ where $x \notin S$, we have that:*

$$\Pr[F(K, x) = F(K_S, x) : K \leftarrow \mathsf{KeyGen}(1^{\lambda}), K_S \leftarrow \mathsf{Puncture}_F(K, S)] = 1.$$

- ***Pseudorandom at punctured points:*** *For every QPT adversary (A_1, A_2), there exists a negligible function* negl *such that the following holds. Consider an experiment where $K \leftarrow \mathsf{KeyGen}_F(1^\lambda)$, $(S, \sigma) \leftarrow A_1(1^\lambda)$, and $K_S \leftarrow \mathsf{Puncture}_F(K, S)$. Then, for all $x \in S$,*

$$\left|\Pr[A_2(\sigma, K_S, S, F(K, x)) = 1] - \Pr_{r \leftarrow \{0,1\}^{m(\lambda)}}[A_2(\sigma, K_S, S, r) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Definition A.10. *A statistically injective (puncturable) PRF family with (negligible) failure probability $\epsilon(\cdot)$ is a (puncturable) PRF family F such that with probability $1 - \epsilon(\lambda)$ over the random choice of key $K \leftarrow \mathsf{KeyGen}_F(1^\lambda)$, we have that $F(K, \cdot)$ is injective.*

We will also utilize extracting PRFs: these are PRFs that are strong extractors on their inputs.

Definition A.11 (Extracting PRF). *An extracting (puncturable) PRF with error $\epsilon(\cdot)$ for min-entropy $k(\cdot)$ is a (puncturable) PRF F mapping $n(\lambda)$ bits to $m(\lambda)$ bits such that for all λ, if X is any distribution over $n(\lambda)$ bits with min-entropy greater than $k(\lambda)$, then the statistical distance between $(K, F(K, X))$ and $(K, r \leftarrow \{0,1\}^{m(\lambda)})$ is at most $\epsilon(\cdot)$, where $K \leftarrow \mathsf{KeyGen}(1^\lambda)$.*

Puncturable PRFs can be obtained by modifying the famous [GGM86] construction, which uses only one-way functions. [SW21] showed that puncturable statistically injective PRFs and extracting puncturable PRFs with the required input-output size can be built from one-way functions as well.

Note that these constructions above can all be made post-quantum [Zha21]. The following theorems from [SW21] thereby hold also against bounded quantum adversaries.

Theorem A.12 ([SW21] Theorem 1, [GGM86]). *If post-quantum one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a post-quantum puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits.*

Theorem A.13 ([SW21] Theorem 2). *If post-quantum one-way functions exist, then for all efficiently computable functions $n(\lambda)$, $m(\lambda)$, and $e(\lambda)$ such that $m(\lambda) \geq 2n(\lambda) + e(\lambda)$, there exists a post-quantum puncturable statistically injective PRF family with failure probability $2^{-e(\lambda)}$ that maps $n(\lambda)$ bits to $m(\lambda)$ bits.*

Theorem A.14 ([SW21] Theorem 3). *If post-quantum one-way functions exist, then for all efficiently computable functions $n(\lambda)$, $m(\lambda)$, $k(\lambda)$, and $e(\lambda)$ such that $n(\lambda) \geq k(\lambda) \geq m(\lambda) + 2e(\lambda) + 2$, there exists a post-quantum extracting puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits with error $2^{-e(\lambda)}$ for min-entropy $k(\lambda)$.*

B Missing Details for the Proof of Anti-Piracy

Here we first prove Claim 3.11, then explain that the proof still holds by replacing PI and TI with $\mathsf{ATI}^{\epsilon,\delta}$ and $\mathsf{TI}^{\epsilon,\delta}$ for some ϵ, δ.

Proof for Claim 3.11. We consider the following distributions $\mathcal{D}_\lambda$ (as in Definition A.3):

- Let $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{CRUD.Setup}(1^\lambda, k)$ where $\mathsf{sk} = (\mathsf{sk}_1, \cdots, \mathsf{sk}_k)$ and $\mathsf{pk} = (\mathsf{pk}_1, \cdots, \mathsf{pk}_k)$. Let ρ_{sk} be the unclonable decryption key. We specify $\mathsf{pk}_j = \{\mathsf{iO}(A_l + s_l), \mathsf{iO}(A_l^\perp + s'_l)\}_{l \in [\ell]}, \mathsf{sk}_j = \{A_l, s_l, s'_l\}_{l \in [\ell]}$.

- Let ρ be the output of $\mathcal{B}$.
- Sample $b \leftarrow \{0,1\}$ and $r \leftarrow \{0,1\}^\ell$.
- Let $\mathsf{Can}_{i,0}(\cdot) = \mathsf{Can}_{A_i}(\cdot)$ and $\mathsf{Can}_{i,1}(\cdot) = \mathsf{Can}_{A_i^\perp}(\cdot)$ where $\mathsf{Can}_{A_i}(\cdot), \mathsf{Can}_{A_i^\perp}(\cdot)$ are the functions defined in Definition 2.3.
- Define function f as follows:

$$f(u_1, \cdots, u_\ell) = \mathsf{Can}_{1,r_1}(u_1) || \cdots || \mathsf{Can}_{\ell,r_\ell}(u_\ell).$$

  Let $s_{i,0} = s_i$ and $s_{i,1} = s_i'$. Let the "lock value" y be the following:

$$y = \mathsf{Can}_{1,r_1}(s_{1,r_1}) || \cdots || \mathsf{Can}_{\ell,r_\ell}(s_{\ell,r_\ell}).$$

  Let $C_{m_b,r}$ be the compute-and-compare program $\mathsf{CC}[f, y, m_b]$.
- Run the obfuscation algorithm $\mathsf{CC.Obf}$ on $C_{m_b,r}$ and obtain the obfuscated program $\tilde{\mathsf{CC}}_{m_b,r} = \mathsf{CC.Obf}(C_{m_b,r})$. Let $\mathsf{ct}_j = \hat{\tilde{\mathsf{CC}}}_{m_b,r} = \mathsf{iO}(\tilde{\mathsf{CC}}_{m_b,r})$. Note this is the ciphertext by encrypting using a public key pk_j on m_b.
- Generate other ciphertext $\mathsf{ct}_1, \cdots, \mathsf{ct}_{j-1}, \mathsf{ct}_{j+1}, \cdots, \mathsf{ct}_k$ using other public keys.
- Let the distribution be

$$(\mathsf{CC}[f, y, m_b], \mathsf{aux}),$$

  where aux is ρ, pk and other ciphertext $\mathsf{ct}_1, \cdots, \mathsf{ct}_{j-1}, \mathsf{ct}_{j+1}, \cdots, \mathsf{ct}_k$.

Consider another distribution $\mathcal{D}'_\lambda$, where it is identical to the above distribution, except $\mathsf{CC}[f, y, m_b]$ is replaced with a simulated program.

By Theorem A.8 and condition (2), if one can notice probability gap with non-negligible probability, $\mathcal{D}_\lambda$ and $\mathcal{D}'_\lambda$ must be distinguishable. Thus, by the security of compute-and-compare obfuscation, there must exist an efficient quantum algorithm that given f, y, aux recovers every vector in Can_{i,r_i} with non-negligible probability. $\square$

Discussing on Using Efficient API and ATI. We first notice that in the proof, to make the first part of argument work, we need to show that with API and ATI, Claim 3.9 still holds. By setting $\epsilon = \gamma/4$ and δ is an exponentially small function, Claim 3.9 still holds with probability at least $1 - O(k\delta)$: by applying API (with all honest generated ciphertexts) on every decryptors, with probability at least $1 - k\delta$, every outcome is greater than $\frac{1}{2} + \frac{3}{4}\gamma$; for any quantum program, by applying API (with all junk ciphertexts) on every decryptors, with probability at least $1 - k\delta$, every outcome is small than $\frac{1}{2} + \frac{1}{4}\gamma$. Thus, we can simply try to identify if there is a probability gap more than $\gamma/(2k)$ instead of γ/k. The rest of the proof goes in a similar way.

C Security Proof for Signature Copy-Protection

C.0.1 Proof of Correctness

It is easy to see that all algorithms in the construction are efficient. We then show that our construction satisfies correctness as defined in Section 4.1.

Lemma C.1. *The above construction satisfies correctness.*

Proof. If for an input x, keys K_2, K_2, the step 1 check criterion in the program P is not satisfied, then the program CPSign outputs $F_1(K_1, \cdot)$ with certainty. The verification will thus pass with certainty as well.

Let us show that for any fixed input $x^* = x_0^*||x_1^*||x_2^*$, only negligible fraction of possible keys K_2, K_3 will let the step 1 check pass.

Suppose there exists an input $x^* = x_0^*||x_1^*||x_2^*$ such that for some inverse polynomial fraction of possible keys K_2, K_3, the step 1 check passes. Let us define $\hat{x}_2^*$ to be the first ℓ_0 bits of x_2^* and $\hat{F}_3(K_3, \cdot)$ be the function that outputs the first ℓ_0 bits of $F_3(K_3, \cdot)$. $\hat{F}_3$ is also a PRF because its output is a truncation of another PRF F_3's output. To pass check the step 1, $(x_0^*, x_1^*, \hat{x}_2^*)$ should satisfy the equation:

$$\hat{F}_3(K_3, x_1^*) \oplus x_0^* = \hat{x}_2^*.$$

By our assumption, for a non-negligible fraction of K_3, the above equation holds. Then we can build a non-uniform algorithm for breaking the security of $\hat{F}_3$ and consequentially, breaking the security of F_3: given oracle access to $\hat{F}_3(K_3, \cdot)$ for a random K_3, or a truly random function $f(\cdot)$, the algorithm simply queries on x_1^* and checks if the output is $x_0^* \oplus \hat{x}_2^*$; if yes, it outputs 1 (indicating the function is $\hat{F}_3(K_3, \cdot)$); otherwise, it outputs 0 (indicating the function is a truly random funtion). Since the above equation holds for some inverse polynomial fraction of K_3, our non-uniform algorithm succeeds with non-negligible probability.

□

Remark C.2. *Non-uniform security of PRFs can be based on non-uniform security of OWFs, the correctness of our construction relies on the existence of non-uniform secure post-quantum OWFs. Such security is provably achievable in the quantum random oracle model[CGLQ20] or should be achieved from non-uniform security of post-quantum candidate assumptions such as LWE.*

C.0.2 Existential Unforgeability

We prove the following theorem:

Theorem C.3. *Assuming the existence of post-quantum secure indistinguishability, one-way functions and puncturable PRFs, the above construction satisfies existential unforgeability.*

Proof. We describe a sequence of hybrids to prove the theorem above.

Hybrid 0 This hybrid corresponds to the original security game.

- The adversary gives the challenger the message x^*.
- The challenger samples $(\mathsf{sk} = (\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign})), \mathsf{vk} = \mathsf{iO}(\mathsf{VK})) \leftarrow \mathsf{Setup}(1^\lambda, k)$. It gives vk to $\mathcal{A}$;
- The adversary queries the signing oracle for a polynomial number of times on messages $x \neq x^*$.
- The adversary provides a signature sig^* for message m^*. The challenger accepts if and only if $\mathsf{Verify}(\mathsf{vk}, x^*, \mathsf{sig}^*) = 1$

Note that the message x^* is sent in before the challenger samples sk, vk. First, x^* hits the hidden trigger with exponentially small probability; even if it does, the challenger can just re-sample the keys.

Hybrid 1 In this hybrid, after the adversary sends in the message x^*, the challenger computes $z^* = \mathsf{OWF}(F_1(K_1, x^*))$, and punctures the PRF key K_1 on x^*:: $K_{1,x^*} = \mathsf{PRF.Puncture}(K_1, x^*)$. Then it generates $\mathsf{vk} = \mathsf{iO}(\mathsf{VK}')$ as in Figure 11.

```
Hardcoded: Keys $K_{1,x^*}, K_2, K_3$
On input $x = x_0||x_1||x_2$ and sig:
  1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
       Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output $Q'(\mathsf{mode} = \mathsf{check}, \mathsf{sig}||0^{\ell_0 \cdot \lambda - m})$, where sig is padded with 0's to the length of $\ell_0 \cdot \lambda$.
  2. Normal Mode:
       (a) If $x = x^*$: check if $\mathsf{OWF}(\mathsf{sig}) = z^*$. If so, output 1. Otherwise, output 0.
       (b) Else check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_{1,x^*}, x))$. If so, output 1. Otherwise, output 0.
```

Figure 11: Program VK'

Programs VK in Hybrid 0 and VK' in Hybrid 1 have the exactly same functionality, by the functionality preserving property of punctured PRF. We can therefore invoke the iO security to argue their indistinguishability.

Hybrid 2 In this hybrid, the challenger replaces $z^* = \mathsf{OWF}(F_1(K_1, x^*))$ with $z^* = \mathsf{OWF}(y)$, where $y \leftarrow \{0,1\}^m$.

Here we invoke puncturable PRF's (selective) pseudorandomness at punctured points property. The reduction gives the signature adversary's x^* to the challenger; receives the punctured key K_{1,x^*}, a value that is either $F_1(K_1, x^*)$ or uniformly random y from the challenger; it then prepares the verification program $\mathsf{iO}(\mathsf{VK}')$. If the adversary successfully forges, then the reduction outputs guess 0 (for real evaluation K_{1,x^*}), otherwise it outputs 1(for uniform random y).

In the end, we show that the adversary's advantage in Hybrid 2 is negligible. Suppose its advantage is non-negligible, then we can build an algorithm that breaks the security of one-way functions. The reduction algorithm receives x^* from the adversary; it also receives the OWF challenge t from the challenger and sets z^* in VK' to be t. If the adversary forges a signature sig^* on x^* that passes verification, then the reduction algorithm can output sig^* as the answer to the OWF challenge since $\mathsf{OWF}(\mathsf{sig}^*) = t$.

□

C.1 Proof of Anti-Piracy Security

In this subsection, we prove the anti-piracy security, Theorem 4.4. As a first step, we give the following lemma from [SW21].

Lemma C.4 (Lemma 1 in [SW21]). *Except with negligible probability over the choice of the key K_2, the following two statements hold:*

1. *For any fixed x_1, there exists at most one pair (x_0, x_2) that will cause the step 1 check in Program P to pass.*
2. *There are at most 2^{ℓ_2} values of x that can cause the step 1 check to pass.*

Proof Overview One important component of the proof is leveraging the sparse hidden triggers in the program CPSign and VK. Intuitively, sampling a unifromly random input is indistinguishable from sampling an element from the sparse hidden trigger set. After we move to a hybrid where the challenge messages are sampled from the hidden trigger set, we will be able to find correspondence between the signature anti-piracy game and the unclonable decryption anti-piracy game. We can then reduce to the security of our unclonable decryption scheme.

Definition C.5 (Hidden Trigger Inputs)**.** *An input x is a hidden trigger input of the program P (defined in Figure 9) if it makes the step 1 check in the program be satisfied.*

We will prove a lemma says that no efficient algorithm, given the quantum key, can distinguish between the following two cases: (i) sample two uniformly random inputs, and (ii) sample two inputs in the hidden trigger set.

Before describing the lemma, we describe an efficient procedure which takes as input an input/output pair for F_1, PRF keys K_2, K_3, descriptions of cosets, and produces a hidden trigger input.

```
Hardcoded: $y; \{A_i, s_i, s'_i\}_{i\in[\ell_0]}; x_0$.
On input mode, $v$:

  1. if mode = eval:
       Parse $v$ as $v_1, \cdots, v_{\ell_0}$. Output $y$ if and only if for every input $v_i$, if $x_{0,i} = 0$, then
       $v_i$ is in $A_i + s_i$ and otherwise it is in $A_i^\perp + s'_i$.
  2. else if mode = check:
       Parse $v$ as sig$||0^{\ell_0\cdot\lambda-m}$ where the first $m$ bits form a string sig. Check if sig $= y$.
     If so, output 1. Otherwise, output 0.
```

Figure 12: Program Q

Definition C.6. *as in The procedure* GenTrigger *takes as input x_0 (of length ℓ_0), y (of length m, where m is the length of the output of F_1), two PRF keys K_2, K_3 and hidden cosets $\{A_i, s_i, s'_i\}_{i\in[\ell_0]}$:*

1. *Let Q be the program (padded to length $\ell_2 - \ell_0$) in figure 12*
2. *$x'_1 \leftarrow F_2(K_2, x_0||Q)$;*
3. *$x'_2 \leftarrow F_3(K_3, x'_1) \oplus (x_0||Q)$.*
4. *Output $x' = x_0||x'_1||x'_2$.*

Note that for any x_0, y, GenTrigger will produce an input x' such that it starts with x_0 and the evaluation of P on input x' and valid vectors $v_1, \cdots, v_{\ell_0}$ is y.

The following lemma says that any efficient algorithm cannot distinguish if it gets two inputs sampled uniformly at random, or two hidden trigger inputs (sampled according to Definition C.6):

Lemma C.7. *Assuming post-quantum* iO *and one-way functions, any efficient QPT algorithm* $\mathcal{A}$ *cannot win the following game with non-negligible advantage:*

- *A challenger samples* $(\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk} = \mathsf{iO}(\mathsf{VK})) \leftarrow \mathsf{Setup}(1^\lambda)$*; then it prepares a quantum program* $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$.
- *The challenger samples a random input* $u \leftarrow [N]$*. Let* $y_u = F_1(K_1, u)$*. Parse the input as* $u = u_0||u_1||u_2$.
  Let $u' \leftarrow \mathsf{GenTrigger}(u_0, y_u, K_2, K_3, \{A_i, s_i, s'_i\}_{i\in[\ell_0]})$.
- *Similarly, it samples a random input* $w \leftarrow [N]$*. Let* $y_w = F_1(K_1, w)$*. Parse the input as* $w = w_0||w_1||w_2$.
  Let $w' \leftarrow \mathsf{GenTrigger}(w_0, y_w, K_2, K_3, \{A_i, s_i, s'_i\}_{i\in[\ell_0]})$.
- *The challenger flips a coin* b*, and sends* (ρ_K, u, w) *or* (ρ_K, u', w') *to* $\mathcal{A}$*, depending on the outcome.* $\mathcal{A}$ *wins if it guesses* b.

Note that the valid signature output by the signing program for message x is actually the PRF evaluation $F_1(K_1, x)$. Therefore, for the simplicity of notations, we will let the challenger compute the PRF evaluations on challenge messages u, w instead of running the copy-protected program.

The lemma above is similar to [CLLZ21] Lemma 7.17, but different in the sense that we additionally provide the adversary with a verification program $\mathsf{iO}(\mathsf{VK})$. We will present the proof to this lemma in Appendix F.

Next, we show that if Lemma C.7 holds, then our construction satisfies anti-piracy security Appendix G.1. After this, to finish the proof, we will only need to prove Lemma C.7. The core of the latter proof is the "hidden trigger" technique used in [SW21], which we will prove in Appendix F.

Proof for Theorem 4.4. We mark the changes between hybrids in red.

Hybrid 0. Hybrid 0 is the original anti-piracy security game.

1. The challenger samples $(\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$; then it prepares a quantum program $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$.
2. $\mathcal{A}$ upon receiving ρ_K, it runs and prepares a pair of (potentially entangled) quantum states $\sigma[R_1], \sigma[R_2]$.
3. The challenger also prepares two inputs u, w as follows:
   - It samples u uniformly at random. Let $y_u = F_1(K_1, u)$.
   - It samples w uniformly at random. Let $y_w = F_1(K_1, w)$.
4. The outcome of the game is 1 if and only if both quantum programs successfully produce y_u and y_w respectively.

Hybrid 1 The difference between Hybrids 0 and 1 corresponds exactly to the two cases that the adversary needs to distinguish between in the game of Lemma C.7.

1. The challenger samples $(\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$; then it prepares a quantum program $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$.
2. $\mathcal{A}$ upon receiving ρ_K, it runs and prepares a pair of (potentially entangled) quantum states $\sigma[R_1], \sigma[R_2]$.

3. The challenger also prepares two inputs u', w' as follows:
   - It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y_u = F_1(K_1, u)$.
     Let $u' \leftarrow \mathsf{GenTrigger}(u_0, y_u, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
   - It samples $w = w_0||w_1||w_2$ uniformly at random. Let $y_w = F_1(K_1, w)$.
     Let $w' \leftarrow \mathsf{GenTrigger}(w_0, y_w, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
4. The outcome of the game is 1 if and only if both quantum programs successfully produce y_u and y_w respectively.

Assume that there exists an adversary that distinguishes Hybrid 0 and 1 with *non-negligible* probability $\epsilon(\lambda)$, then these exists an adversary that breaks the game in Lemma C.7 with probability $\epsilon(\lambda) - \mathsf{negl}(\lambda)$.

The reduction algorithm receives ρ_k and u, w or u', w' from the challenger in Lemma C.7; it computes y_u, y_w using $\mathsf{iO}(P)$ on the received inputs respectively and gives them to the quantum decryptor states $\sigma[R_1], \sigma[R_2]$. If they both decrypt correctly, then the reduction outputs 0 (i.e. it guess that sampling was uniform), otherwise it outputs 1 (i.e. it guesses that hidden trigger inputs were sampled).

Hybrid 2. In this hybrid, the signatures for challenge messages, $F_1(K_1, u)$ and $F_1(K_1, w)$, are replaced with truly random strings.

1. The challenger samples $(\mathsf{sk} = (\{A_i, s_i, s_i'\}_{i\in[\ell_0]}, K_1, K_2, K_3), \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$; then it prepares a quantum program $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$.
   Note that here CPSign hardcodes K_1, K_2, K_3.
2. $\mathcal{A}$ upon receiving ρ_{sk}, it runs and prepares a pair of quantum states $\sigma[R_1], \sigma[R_2]$.
3. The challenger also prepares two inputs u', w' as follows:
   - It samples u_0 uniformly at random. It then samples a uniformly random y_u.
     Let $u' \leftarrow \mathsf{GenTrigger}(u_0, y_u, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
   - It samples w_0 uniformly at random. It then samples a uniformly random y_w.
     Let $w' \leftarrow \mathsf{GenTrigger}(w_0, y_w, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
4. The outcome of the game is 1 if and only if both quantum programs successfully produce y_u and y_w respectively.

Since $u_0 \neq w_0$ with overwhelming probability and both u_0, w_0 have enough min-entropy $\ell_1 + \ell_2 \geq m + 2\lambda + 4$ (as $u_1||u_2$ and $w_1||w_2$ are completely uniform and not given to the adversary) and F_1 is an extracting puncturable PRF, both outcomes y_u, y_w are statistically close to independently random outcomes. Therefore, Hybrid 1 and Hybrid 2 are statistically indistinguishable.

Hybrid 3. In this hybrid, we only change the order of sampling.

1. The challenger first samples $\{A_i, s_i, s_i'\}_{i\in[\ell_0]}$ and prepares the quantum states $\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}$. It treats the the quantum states $\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}$ as the quantum decryption key ρ_{sk} for our unclonable decryption scheme and the secret key sk is $\{A_i, s_i, s_i'\}_{i\in[\ell_0]}$. Similarly, let $\mathsf{pk} = \{R_i^0, R_i^1\}_{i\in[\ell_0]}$ where $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s_i')$.
2. It samples y_u, y_w uniformly at random. Let $(u_0, Q_0) \leftarrow \mathsf{UD.Enc}(\mathsf{pk}, y_u)$ and $(w_0, Q_1) \leftarrow \mathsf{UD.Enc}(\mathsf{pk}, y_w)$ where $\mathsf{UD.Enc}(\mathsf{pk}, \cdot)$ is the encryption algorithm of the unclonable decryption scheme.

3. The challenger sets $\rho_K = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$ as well as verification key $\mathsf{iO}(\mathsf{VK})$.
4. $\mathcal{A}$ upon receiving ρ_K, it runs and prepares a pair of quantum states $\sigma[R_1], \sigma[R_2]$.
5. The challenger also prepares two inputs u', w' as follows (as $\mathsf{GenTrigger}$ does):
    - Let $u'_1 \leftarrow F_2(K_2, u_0||Q_0)$ and $u'_2 \leftarrow F_3(K_3, u'_1) \oplus (u_0||Q_0)$. Let $u' = u_0||u'_1||u'_2$.
    - Let $w'_1 \leftarrow F_2(K_2, w_0||Q_1)$ and $w'_2 \leftarrow F_3(K_3, w'_1) \oplus (w_0||Q_1)$. Let $w' = w_0||w'_1||w'_2$.
6. The outcome of the game is 1 if and only if both quantum programs successfully produce y_u and y_w respectively.

Note that the only differences of Hybrids 2 and 3 are the orders of executions. Therefore, $\mathcal{A}$ has the same advantage as in Hybrid 2. We would like the highlighted components in the above game to match the unclonable encryption scheme in Figure 4.

However, we would notice that the ciphertexts (u_0, Q_0) and (u_1, Q_1) prepared in Hybrid 3 are in fact different from the ciphertexts in unclonable encryption scheme in the previous section's constructiion, Figure 4. What we need here is a ciphertext that has the same functionality of program Q in Figure 12.

We therefore turn to an unclonable decryption scheme UD' with slightly stronger security: the challenger provides the adversary with iO of the following program as an encryption of message y:

Plaintext: y.
Hardcoded: $\{R_i^0, R_i^1\}_{i\in[\ell_0]}; x_0$.
On input mode, v:

```
1. if mode = eval:
     Parse $v$ as $v_1, \cdots, v_{\ell_0}$. Output $y$ if and only if for every input $v_i$, $R_i^{x_{0,i}}(v_i) = 1$,
   where $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$
2. else if mode = check:
     Parse $v$ as $\mathsf{sig}||0^{\ell_0\cdot\lambda - m}$ where the first $m$ bits form a string sig. Check if $\mathsf{sig} = y$.
   If so, output 1. Otherwise, output 0.
```

Figure 13: Encryption of plaintext y in UD', which is another form of program Q in fig. 12
.

Proposition C.8. *The modified unclonable encryption scheme* UD' *satisfies the (1 collusion resistant) anti-piracy security as defined in Definition 3.4.*

The intuition behind the proposition is as follows: the ciphertext in fig. 13 can be seen as two programs. The first one is the original ciphertext in the fig. 4 scheme; the second program is an obfuscation for a point function that on input sig, checks if it equals y. The second program can be seen as an additional auxiliary information provided to the adversary $\mathcal{A}$ in the anti-piracy game, but can be eventually removed and be of no help to $\mathcal{A}$.

The full proof is more involved, but naturally follows from the anti-piracy security proof in [CLLZ21]. We will give a proof for the proposition above and point out its differences from [CLLZ21] in Appendix D.

Reduction to Unclonable Decryption The next step is to show that if an algorithm $\mathcal{A}$ that wins the game in Hybrid 3 with non-negligible probability $\gamma(\lambda)$, we can build another algorithm $\mathcal{B}$ that breaks the (regular) γ-anti-piracy security with random challenge plaintexts of the underlying unclonable decryption scheme .

- $\mathcal{B}$ plays as the challenger in the game of Hybrid 3.
- $\mathcal{B}$ receives $\rho_{\mathsf{sk}} = \{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}$ and $\mathsf{pk} = \{\mathsf{iO}(A_i + s_i), \mathsf{iO}(A_i^\perp + s'_i)\}_{i\in[\ell_0]}$ in the anti-piracy game of unclonable decryption.
- $\mathcal{B}$ samples PRF keys K_1, K_2, K_3 and a one-way function OWF. Let $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s'_i}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$ as well as $\mathsf{vk} = \mathsf{iO}(\mathsf{VK})$. Here $\mathcal{B}$ can prepare the program CPSign and VK using R_i^0, R_i^1 for $i \in [k]$ provided by the unclonable decryption challenger.
- $\mathcal{B}$ gives ρ_{sk} to $\mathcal{A}$, and $\mathcal{A}$ the produces a pair of quantum states $\sigma[R_1], \sigma[R_2]$.
- $\mathcal{B}$ outputs the decryptors $(\sigma[R_1], \mathsf{P}_1)$ and $(\sigma[R_2], \mathsf{P}_2)$ to the unclonable decryption challenger. P_1 and P_2 are programs that correspond to first running GenTrigger procedures and then evaluating the pirate programs on GenTrigger's output values:
  On input $(\rho_1, \mathsf{ct}_1 = (u_0, Q_1))$ and $(\rho_2, \mathsf{ct}_2 = (w_0, Q_2))$ respectively (where ct_1 and ct_2 represent encryptions of random messages y_u and y_w, chosen by the challenger), P_1 and P_2 behave as follows:
  - P_1: Let $u'_1 \leftarrow F_2(K_2, u_0||Q_0)$ and $u'_2 \leftarrow F_3(K_3, u'_1) \oplus (u_0||Q_0)$. Let $u' = u_0||u'_1||u'_2$. Run (ρ_1, U_1) on u'.
  - P_2: Let $w'_1 \leftarrow F_2(K_2, w_0||Q_1)$ and $w'_2 \leftarrow F_3(K_3, w'_1) \oplus (w_0||Q_1)$. Let $w' = w_0||w'_1||w'_2$. Run (ρ_2, U_2) on w' respectively.

If $\mathcal{A}$ succeeds in the game of Hybrid 3, its pirate programs would output the "correct signatures" y_u, y_w for messages u', w'. As we can see, they are also correct decryption outputs for ciphertexts ct_1 and ct_2.

Therefore, the programs prepared by $\mathcal{B}$ will successfully decrypt the encryptions of uniformly random plaintexts. Thus, $\mathcal{B}$ breaks γ-anti-piracy security with random challenge plaintexts. □

D Proof for Proposition C.8

Proof. The proof follows from the arguments of anti-piracy security for unclonable decryption in [CLLZ21] section 6.4, except providing the adversary with some additional auxiliary information. We thereby omit some repetitive details and refer the reader to the full proof in the above paper.

More specifically, we can view the ciphertext in Figure 13 as two separate programs:

1. The eval mode program is iO of a program that hardcodes $\{R_i^0, R_i^1\}_{i\in[\ell_0]}, \mathsf{msg}, x$. It does the following: [9]
   On input v: Parse v as $v_1, \cdots, v_{\ell_0}$. Output msg if and only if for every input v_i, $R_i^{x_{0,i}}(v_i) = 1$, where $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$.
2. The check mode program is an iO of a point function:
   on input v: Parse v as $\mathsf{sig}||0^{\ell_0\cdot\lambda-m}$ where the first m bits form a string sig. Check if $\mathsf{sig} = \mathsf{msg}$. If so, output 1. Otherwise, output 0.

[9]To avoid confusion of notations, we denote the message y in the same programs in Section 4 as msg here.

The eval mode program is exactly the ciphertext generated in the original construction Figure 4. The check program is an auxiliary information given to the adversary.

By the security definition definition 3.4, a successful adversary produces pirate programs $\sigma[R_1], \sigma[R_2]$ that satisfy $\mathrm{Tr}[(\mathsf{TI}_{1/2+\gamma}(\mathcal{P}_{\mathcal{D}}) \otimes \mathsf{TI}_{1/2+\gamma}(\mathcal{P}_{\mathcal{D}}))\sigma] \geq$ non-negl(λ). The testing distribution $\mathcal{D}$ is a distribution for sampling a challenge ciphertext by preparing the two programs above, for some uniformly random x.

We know by theorem 2.7 that now if we apply the measurements $\mathsf{TI}_{1/2+\gamma}(\mathcal{P}_{\mathcal{D}})$ again to the states after measurements respectively, the probability that their outcomes are both 1 is $(1 - \mathsf{negl}(\lambda))$. Let us denote these states as $\sigma[R_1]', \sigma[R_2]'$.

Next, we switch to a different testing distribution $\mathcal{D}'$ of sampling challenge ciphertexts:

- The eval mode program is prepared as follows:
  - Let $\mathsf{Can}_{i,0}(\cdot) = \mathsf{Can}_{A_i}(\cdot)$ and $\mathsf{Can}_{i,1}(\cdot) = \mathsf{Can}_{A_i^\perp}(\cdot)$. $\mathsf{Can}_{A_i}(\cdot)$ denotes a function that computes the lexicographically smallest vector contained in $A_i + s_j$. Likewise for $\mathsf{Can}_{A_i^\perp}(\cdot)$.
  - Define function f as follows:
  $$f(u_1, \cdots, u_\ell) = \mathsf{Can}_{1,x_1}(u_1)|| \cdots ||\mathsf{Can}_{\ell,r_\ell}(u_\ell).$$
  Let $s_{i,0} = s_i$ and $s_{i,1} = s_i'$. Let the "lock value" lock be the following:
  $$\mathsf{lock} = \mathsf{Can}_{1,x_1}(s_{1,rx1})|| \cdots ||\mathsf{Can}_{\ell,r_\ell}(s_{\ell,r_\ell}).$$
  Let $C_{\mathsf{msg},x}$ be the compute-and-compare program $\mathsf{CC}[f, \mathsf{lock}, \mathsf{msg}]$.
  - Run the obfuscation algorithm $\mathsf{CC.Obf}$ on $C_{\mathsf{msg},r}$ and obtain the obfuscated program $\mathsf{CC.Obf}_{m,x} = \mathsf{CC.Obf}(C_{\mathsf{msg},x})$. Let $\hat{\mathsf{CC}}_{\mathsf{msg},r} = \mathsf{iO}(\mathsf{CC.Obf}_{\mathsf{msg},x})$.
  - Let the ciphertext be $(\hat{\mathsf{CC}}_{\mathsf{msg},r}, x)$.
- the check mode program stays unchanged.

Since the eval mode programs prepared in $\mathcal{D}$ and $\mathcal{D}'$ have exactly the same functionality, we can see that distribution $\mathcal{D}'$ is computationally indistinguishable from $\mathcal{D}$ by the post quantum security of iO. Therefore, by Theorem A.8, states σ' will still pass the test of $\mathsf{TI}_{1/2+\gamma}(\mathcal{P}_{\mathcal{D}'}) \otimes \mathsf{TI}_{1/2+\gamma}(\mathcal{P}_{\mathcal{D}'})$ with $(1 - \mathsf{negl}(\lambda))$ probability.

Next, we switch the ciphertext from $\mathsf{iO}(\mathsf{CC.Obf}_{\mathsf{msg},x})$ to $\mathsf{iO}(\mathsf{Sim})$ where Sim is a simulated program that always output $\perp$. Let us denote this distribution as $\mathcal{D}''$ and it is clear that no pirate program can decrypt with success probability larger than $1/2$ under distribution $\mathcal{D}''$.

We then invoke the security of compute-and-compare obfuscation. The lock value lock should be unpredictable even if given description of f and msg, as long as lock is chosen independent of msg and f. Our f, msg and lock satisfy such a condition. In this case, the additional check mode program is of no use to the compute-and-compare adversary: even if the point function inside the program is given in the plain to him, he should not be able to distinguish $\mathcal{D}''$ and $\mathcal{D}'$.

However, the pirate programs have a non-negligible difference in success probabilities when running under $\mathcal{D}''$ and $\mathcal{D}'$. By the contrapositive of definition A.3, if there exists an adversary that distinguishes between $(\mathcal{D}', \mathsf{aux} = (f, \mathsf{msg}, \sigma'))$ and $(\mathcal{D}'', \mathsf{aux} = (f, \mathsf{msg}, \sigma'))$, then there exists an extractor that predicts the lock value lock. The rest of the proof is a delicate argument on how to extract the lock values, which should follow exactly from [CLLZ21] section 6.4.

□

E Bounded Collusion Resistant Copy-Protection for Signatures

Let us denote our underlying 1 collusion resistant signature scheme as SigCP. We give the following construction for bounded collusion resistant copy-protection for signatures $\mathsf{CRSigCP}$. Note that the following construction is not a generic transformation from any signature copy-protection scheme to a collusion resistant one, but takes use of certain structures specific to our scheme.

$\mathsf{CRSigCP.Setup}(1^\lambda, k)$:

- Run $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SigCP.Setup}(1^\lambda)$.
- Prepare a program CRVK that takes inputs (x, sig, i) and outputs $0/1$ as in fig. 16.
- Let $\mathsf{vk} = \mathsf{iO}(\mathsf{CRVK})$. Output $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{CRSigCP.QKeyGen}(\mathsf{sk})$: Let us first slightly revise the underlying algorithm $\mathsf{SigCP.QKeyGen}(\mathsf{sk})$ such that the program CPSign is generated as in fig. 15

- For $i \in [k]$, $\rho_i \leftarrow \mathsf{SigCP.QKeyGen}(\mathsf{sk})$.
- Let $\rho_{\mathsf{sk},i}$ be ρ_i padded with a classical index i, i.e., $\rho_{\mathsf{sk},i} = \rho_i \otimes |i\rangle\langle i|$.
- The classical part of ρ_i is iO of a program CPSign_i.
  Compile them into a program CRSign: on input (ρ_i, x, i), CRSign will run $\mathsf{iO}(\mathsf{CPSign}_i)$ on inputs (ρ_i, x, i) according to index i.
- Output $\rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k}$ and $\mathsf{iO}(\mathsf{CRSign})$.

$\mathsf{CRSigCP.Sign}(\rho_{\mathsf{sk}}, x)$:

- Parse ρ_{sk} as ρ_i, index i as well as program $\mathsf{iO}(\mathsf{CRVK})$.
- Output $\mathsf{sig} = (\mathsf{sig}_i \leftarrow \mathsf{iO}(\mathsf{CRSign})(\rho_i, x, i), i)$.

$\mathsf{CRSigCP.Verify}(\mathsf{vk}, x, \mathsf{sig})$:

- Parse $\mathsf{vk} = \mathsf{iO}(\mathsf{CRVK})$ and parse sig as a signature sig_i and an index i.
- Output the result of running $\mathsf{iO}(\mathsf{CRVK})$ on input (x, sig_i, i).

Figure 14: Collusion Resistant Copy-Protection for Signature Scheme.

Correctness Note that according to the underlying algorithm $\mathsf{SigCP.Setup}$ and $\mathsf{SigCP.QKeyGen}$, in each copy CPSign_i for $i \in [k]$: the keys K_1, K_2, K_3 are identical; but the subspace membership functions $\{\{R_j^0, R_j^1\}_{j\in[\ell_0]}\}_{i\in[k]}$ correspond to $\ell_0 \cdot k$ number of i.i.d. generated shifted subspaces.

The correctness easily follows from the correctness of the single-copy scheme: each copy of the secret key and corresponding verification key are generated independently; the signing and verification algorithm will use the i-th signing key/verification key as instructed in the input.

Remark E.1. *The above construction is an almost black-box translation from the* 1*- bounded collusion scheme. We therefore give up some efficiency for the sake of clarity. To provide more efficiency, we can simply generate* CPSign_i *inside* $\mathsf{CRSigCP.QKeyGen}$ *on its own without running an underlying* $\mathsf{SigCP.QKeyGen}$*, so that we have less nested obfuscation.*

Hardcoded: Keys K_1, K_2, K_3, R_j^0, R_j^1 for all $j \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$ and index i:

```
1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
      Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output
   $Q'(\mathsf{mode} = \mathsf{eval}, v_1, \cdots, v_{\ell_0}, i)$.
2. Otherwise, check if the following holds: for all $j \in [\ell_0]$, $R_j^{x_{0,j}}(v_j) = 1$ (where $x_{0,j}$
   is the $j$-th bit of $x_0$).
      Normal Mode: If so, output $F_1(K_1, x)$. Otherwise, output $\perp$.
```

Figure 15: Program CPSign Revised: the circuit Q' now takes in additional input, index i. Note that we change the indices for the subspaces to $j \in [\ell_0]$ in order to distinguish from the index $i \in [k]$, which specifies the i-th copy of program.

Hardcoded: Keys K_1, K_2, K_3.
On input $x = x_0||x_1||x_2$, sig and i:

```
1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
      Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output $Q'(\mathsf{mode} =$
   $\mathsf{check}, \mathsf{sig}||0^{\ell_0 \cdot \lambda - m}, i)$, where sig is padded with 0's to the length of $\ell_0 \cdot \lambda$ and $i$ is an
   index in $[k]$.
2. Otherwise:
      Normal Mode: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Other-
   wise, output 0.
```

Figure 16: Program CRVK

Collusion Resistant Anti-Piracy Security Our security proof will follow the hybrids in the 1-bounded collusion resistant setting: hybrid 0 is the original anti-piracy game; in hybrid 1, we change the challenge messages $x_1, \cdots, x_k$ from uniformly random to outputs of a hidden trigger algorithm; in hybrid 2, we change the PRF evaluations $y_i = F_1(K_1, x_1)$ to uniformly random values y_i' for each $i \in [k]$. In hybrid 3, we re-order the sampling procedures to match the security game of unclonable decryption. Finally, we can formulate a reduction to the security of the underlying k collusion resistant unclonable decryption scheme UD'.

One would observe that two things may prevent us from directly generalizing the 1-bounded collusion proof: First, to argue indistinguishability of hybrid 0 and hybrid 1, we now rely on a k-hidden trigger inputs version of Lemma C.7. As we will state and prove in lemma F.3, we can obtain such a generalized lemma by exploiting the construction of our collusion resistant signature copy-protection scheme.

Second, we need to argue that the underlying modified unclonable decryption UD' satifies k collusion resistance. As we have seen, in the 1 collusion resistance case, UD' has the same structure and security proof (by leveraging the property of compute-and-compare obfuscation) as the construction in fig. 4. Therefore, UD' satisfies k collusion resistance with the general translation proved in the main body of this paper.

F Proof for Lemma C.7 and k-Hidden Trigger Lemma

This lemma states that a QPT adversary can not distinguish a pair of uniformly inputs from a pair of hidden trigger inputs.

For the clarity of presentation, we show the following lemma about the indistinguishability of a single random input or a single hidden trigger input. Afterwards, we will discuss how the proof for Lemma F.1 can translate to a proof for Lemma C.7 as well as a proof for the k-Hiden Trigger lemma.

However, we can not get Lemma C.7 by simply applying Lemma F.1 twice (or likewise, k times), we will elaborate the generalization of the proof.

Lemma F.1. *Assuming post-quantum* iO *and one-way functions, for every efficient QPT algorithm* $\mathcal{A}$*, it can not distinguish the following two cases with non-negligible advantage:*

- *A challenger samples* $(\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk} = \mathsf{iO}(\mathsf{VK})) \leftarrow \mathsf{Setup}(1^\lambda)$*; then it prepares a quantum program* $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{CPSign}))$.
- *It samples a random input* $u \leftarrow [N]$*. Let* $y = F_1(K_1, u)$*. Parse the input as* $u = u_0||u_1||u_2$.
- *Let* $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
- *It flips a coin* b *and outputs* (ρ_K, u) *or* (ρ_K, u') *depending on the coin.*

Note that we will mark the changes between the current hybrid and the previous hybrid in red.

Proof of Lemma F.1 . The proof mostly follows from the proof for 7.17 in [CLLZ21], with differences in the programs we apply puncturing on.

Note that the program CPSign (fig. 9) and program VK(fig. 10) are highly similar in their structures and share all the hardcoded values. Therefore, for the simplicity of notations, we combine these two programs into one in the following proof: in the eval mode, it functions as CPSign; in the check mode, it functions as VK. It is easy to see that the overall functionalities will be exactly the same as the original construction.

Hybrid 0. This is the original game where the input is sampled either uniformly at random or sampled as a hidden triggers input.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s_i' for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s_i')$ (padded to the length upper bound $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s_i'}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y = F_1(K_1, u)$.
4. It generates $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$.
5. Generate the program P as in Figure 17. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin b.

Hardcoded: Keys K_1, K_2, K_3, R_i^0, R_i^1 for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$, vectors $v_1, \cdots, v_{\ell_0}$, and a string mode:

- If $\mathsf{mode} = \mathsf{eval}$:
  1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
     Treat Q' as a circuit and outputs $Q'(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
  2. Otherwise, check if the following holds: for all $i \in [\ell_0]$, $R^{x_{0,i}}(v_i) = 1$.
     If they all hold, output $F_1(K_1, x)$. Otherwise, output $\perp$.
- If $\mathsf{mode} = \mathsf{check}$: Parse vectors $v_1, \cdots, v_{\ell_0}$ as $\mathsf{sig}||0^{\ell_0\cdot\lambda - m}$).
  1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
     Treat Q' as a (classical) circuit and output $Q'(\mathsf{mode} = \mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda - m})$.
  2. Otherwise: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Otherwise, output 0.

Figure 17: Program CPSign and VK combined

Hybrid 1 In this hybrid, the key K_1 in the program P is punctured at u, u'. The indistinguishability of Hybrid 0 and Hybrid 1 comes from the security of indistinguishability obfuscation.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s'_i for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s'_i}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y = F_1(K_1, u)$.
4. It samples $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s'_i\}_{i\in[\ell_0]})$. Let Q be the obfuscation program during the execution of $\mathsf{GenTrigger}$.
5. Generate the program as in Figure 18. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

Hardcoded: Constants u, u'; Keys $K_1 \setminus \{u, u'\}, K_2, K_3, R_i^0, R_i^1$ for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$ and a string mode:

- If mode $=$ eval :
  1. If $x = u$ or u', output $Q(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
  2. If $F_3(K_3, x_1) \oplus x_2 = x'_0||Q'$ and $x_0 = x'_0$ and $x_1 = F_2(K_2, x'_0||Q')$:
     Treat Q' as a circuit and outputs $Q'(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
  3. Otherwise, it checks if the following holds: for all $i \in [\ell_0]$, $R^{x_{0,i}}(v_i) = 1$.
     If they all hold, output $F_1(K_1, x)$. Otherwise, outputs $\perp$.
- If mode $=$ check : Parse vectors $v_1, \cdots, v_{\ell_0}$ as $\mathsf{sig}||0^{\ell_0\cdot\lambda-m}$.
  1. If $x = u$ or u', it outputs $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
  2. If $F_3(K_3, x_1) \oplus x_2 = x'_0||Q'$ and $x_0 = x'_0$ and $x_1 = F_2(K_2, x'_0||Q')$:
     Treat Q' as a (classical) circuit and output $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
  3. Otherwise: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Otherwise, output 0.

Figure 18: Program VK and CPSign combined for Hybrid 1

Note that starting from this hybrid, whenever we mention K_1 *inside a program* P, we mean to use the punctured key $K_1 \setminus \{u, u'\}$. Similar notations of punctured keys K_2, K_3 inside other programs will appear in the upcoming hybrids.

Hybrid 2. In this hybrid, the value of $F_1(K_1, u)$ is replaced with a uniformly random output. The indistinguishability of Hybrid 1 and Hybrid 2 comes from the pseudorandomness at punctured points of a puncturable PRF.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s'_i for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s'_i}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y \leftarrow [M]$.
4. It samples $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s'_i\}_{i\in[\ell_0]})$. Let Q be the obfuscation program during the execution of $\mathsf{GenTrigger}$.
5. Generate the program P as in Figure 18. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

Hybrid 3. In this hybrid, the check on the second line will be skipped if x_1 is equal to u_1 or u_1'. By Lemma 2 of [SW21], adding this check does not affect its functionality, except with negligible probability.

The lemma says, to skip the check on the second line, x_1 will be equal to one of $\{u_1, u_1'\}$. To see why it does not change the functionality of the program, by Lemma C.4 and for all but negligible fraction of all keys K_2, if $x_1 = u_1'$, there is only one way to make the check satisfied and the input is u_0, u_2'. This input $u' = u_0||u_1'||u_2'$ is already handled in the first line. Therefore, the functionality does not change.

After this change, $F_3(K_3, \cdot)$ will never be executed on those inputs. We can then puncture the key K_3 on them. The indistinguishability comes from the security of iO.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s_i' for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s_i')$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s_i'}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y \leftarrow [M]$.
4. It samples $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$. Let Q be the obfuscation program during the execution of $\mathsf{GenTrigger}$.
5. Generate the program as in Figure 19. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

```
Hardcoded: Constants $u, u'$; Keys $K_1 \setminus \{u, u'\}, K_2, K_3 \setminus \{u_1, u_1'\}, R_i^0, R_i^1$ for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$ and a string mode:

  • If mode = eval :
      1. If $x = u$ or $u'$, output $Q(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
      2. If $x_1 = u_1$ or $u_1'$, skip this check. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and
         $x_1 = F_2(K_2, x_0'||Q')$:
            Treat $Q'$ as a circuit and outputs $Q'(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
      3. Otherwise, it checks if the following holds: for all $i \in [\ell_0]$, $R^{x_{0,i}}(v_i) = 1$.
            If they all hold, output $F_1(K_1, x)$. Otherwise, outputs $\perp$.

  • If mode = check : Parse vectors $v_1, \cdots, v_{\ell_0}$ as $\mathsf{sig}||0^{\ell_0\cdot\lambda-m}$.
      1. If $x = u$ or $u'$, it outputs $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
      2. If $x_1 = u_1$ or $u_1'$, skip this check. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and
         $x_1 = F_2(K_2, x_0'||Q')$:
            Treat $Q'$ as a (classical) circuit and output $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
      3. Otherwise: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Otherwise,
         output 0.
```

Figure 19: Program CPSign and VK combined in Hybrid 3

Hybrid 4. In this hybrid, before checking $x_1 = F_2(K_2, x_0'||Q')$, it checks if $x_0'||Q' \neq u_0||Q$. Because if $x_0'||Q' = u_0||Q$ and the last check $x_1 = F_2(K_2, x_0'||Q')$ is also satisfied, we know that

$$x_1 = F_2(K_2, x_0'||Q') = F_2(K_2, u_0||Q) = u_1' \text{ (by the definition of } \mathsf{GenTrigger}).$$

Therefore the step 2 will be skipped (by the first check). Thus, we can puncture K_2 at $u_0||Q$ The indistinguishability also comes from the security of iO.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s_i' for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s_i')$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s_i'}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y \leftarrow [M]$.
4. It samples $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s_i'\}_{i\in[\ell_0]})$. Let Q be the obfuscation program during the execution of GenTrigger.
5. Generate the program P as in Figure 20. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

Hardcoded: Constants u, u'; Keys $K_1 \setminus \{u, u'\}$, $K_2 \setminus \{u_0||Q\}$, $K_3 \setminus \{u_1, u_1'\}$, R_i^0, R_i^1 for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$ and a string mode:

- If $\mathsf{mode} = \mathsf{eval}$:
  1. If $x = u$ or u', it outputs $Q(v_1, \cdots, v_{\ell_0})$.
  2. If $x_1 = u_1$ or u_1', skip this check. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$, and $x_0'||Q' \neq u_0||Q$, and $x_1 = F_2(K_2, x_0'||Q')$:
     Treat Q' as a circuit and outputs $Q'(\mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
  3. Otherwise, it checks if the following holds: for all $i \in [\ell_0]$, $R^{x_{0,i}}(v_i) = 1$.
     If they all hold, outputs $F_1(K_1, x)$. Otherwise, outputs $\perp$.
- If $\mathsf{mode} = \mathsf{check}$: Parse vectors $v_1, \cdots, v_{\ell_0}$ as $\mathsf{sig}||0^{\ell_0\cdot\lambda-m}$.
  1. If $x = u$ or u', it outputs $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
  2. If $x_1 = u_1$ or u_1', skip this check. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$, and $x_0'||Q' \neq u_0||Q$, and $x_1 = F_2(K_2, x_0'||Q')$:
     Treat Q' as a (classical) circuit and output $Q'(\mathsf{check}, \mathsf{sig}||0^{\ell_0\cdot\lambda-m})$.
  3. Otherwise: Check if $\mathsf{OWF}(\mathsf{sig}) = \mathsf{OWF}(F_1(K_1, x))$. If so, output 1. Otherwise, output 0.

Figure 20: Program CPSign and VK combined in Hybrid 4

Hybrid 5. In this hybrid, since the key K_2 has been punctured at $u_0||Q$, we can replace the evaluation of $F_2(K_2, \cdot)$ at the input with a uniformly random value. The indistinguishability comes from the pseudorandomness at punctured points of PRF F_2.

We expand the $\mathsf{GenTrigger}$ procedure.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s'_i for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R_i^0 = \mathsf{iO}(A_i + s_i)$ and $R_i^1 = \mathsf{iO}(A_i^\perp + s'_i)$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s'_i}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y \leftarrow [M]$.
4. It samples $u' \leftarrow \mathsf{GenTrigger}(u_0, y, K_2, K_3, \{A_i, s_i, s'_i\}_{i \in [\ell_0]})$ as follows:
    (a) Let Q be the obfuscation of the program (padded to length $\ell_2 - \ell_0$) that takes inputs $v_1, \cdots, v_{\ell_0}$ and outputs y if and only if for every input v_i, if $u_{0,i} = 0$, then v_i is in $A_i + s_i$ and otherwise it is in $A_i^\perp + s'_i$.
    (b) $u'_1 \leftarrow [2^{\ell_1}]$ (since $F_2(K_2, u_0||Q)$ has been replaced with a uniformly random value).
    (c) $u'_2 \leftarrow F_3(K_3, u'_1) \oplus (u_0||Q)$.
    (d) It outputs $u' = u_0||u'_1||u'_2$.
5. Generate the program as in Figure 20. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

Hybrid 6. In this hybrid, since the key K_3 has been punctured at u'_1, we can replace the evaluation of $F_3(K_3, \cdot)$ at u'_1 with a uniformly random value. The indistinguishability comes from the pseudorandomness at punctured points of PRF F_3.

1. It samples random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s'_i for $i = 1, 2, \cdots, \ell_0$. It then prepares programs $R^0_i = \mathsf{iO}(A_i + s_i)$ and $R^1_i = \mathsf{iO}(A^\perp_i + s'_i)$ (padded to length $\ell_2 - \ell_0$). It prepares the quantum state $|\psi\rangle = \bigotimes_i |A_{i,s_i,s'_i}\rangle$.
2. It then samples keys K_1, K_2, K_3 for F_1, F_2, F_3.
3. It samples $u = u_0||u_1||u_2$ uniformly at random. Let $y \leftarrow [M]$.
4. It samples u' as follows:
   (a) $u'_1 \leftarrow [2^{\ell_1}]$;
   (b) $u'_2 \leftarrow [2^{\ell_2}]$.
   (c) It outputs $u' = u_0||u'_1||u'_2$.
5. Generate the program as in Figure 19. The adversary is given $(|\psi\rangle, \mathsf{iO}(P))$ and then u or u' depending on a random coin.

In this hybrids, u, u' are sampled independently, uniformly at random and they are symmetric in the program. The distributions for $b = 0$ and $b = 1$ are identical and even unbounded adversary can not distinguish these two cases. Therefore we finish the proof for Lemma F.1.

Remark F.2. *The program P depends on Q_u. Although Q_u is indexed by u, it only depends on u_0. Thus, the distributions for $b = 0$ and $b = 1$ are identical*

□

Proof for Lemma C.7. As we have briefly mentioned, one can not get Lemma C.7 by simply applying Lemma F.1 twice, as one can not sample a random hidden trigger input by only given the public information in the security game ($\mathsf{GenTrigger}$ requires knowing K_2, K_3), which is essentially required. But the translation is straightforward.

The only difference between Lemma C.7 and Lemma F.1 is the number of inputs sampled: either a single input u (or u') or a pair of independent inputs u, w (or u', w').

All hybrids for Lemma C.7 are the same for the corresponding hybrids for Lemma F.1, except two inputs are sampled. Thus every time K_1, K_2 or K_3 are punctured according to u or u' in the proof of Lemma F.1, K_1, K_2 or K_3 are punctured *twice* according to both u, u' and w, w' in the proof of Lemma C.7.

k-Hidden Trigger Lemma and k-Bounded Collusion Resistant Copy-Protection for Signatures The following lemma says that any efficient algorithm cannot distinguish if it gets k inputs sampled uniformly at random, or k hidden trigger inputs (sampled according to Definition C.6), *but with a modified program Q as in Figure 21.*

This lemma would finish the proof for k-bounded collusion resistant copy-protection for signatures in Appendix E.

Lemma F.3. *Assuming post-quantum* iO *and one-way functions, any efficient QPT algorithm $\mathcal{A}$ cannot win the following game with non-negligible advantage:*

- *A challenger runs* $(\mathsf{sk} = (K_1, K_2, K_3), \mathsf{vk} = \mathsf{iO}(\mathsf{CRVK})) \leftarrow \mathsf{CRSigCP.Setup}(1^\lambda)$*; then it prepares a quantum program* $\rho_{\mathsf{sk}} = (\{\{|A_{j,s_j,s'_j}\rangle\}_{j\in[\ell_0]}\}_{i\in[k]}, \mathsf{iO}(\mathsf{CRSign}))$.
- *For* $i \in [k]$*:*
  the challenger samples a random input $u_i \leftarrow [N]$*. Let* $y_i = F_1(K_1, u_i)$*. Parse* $u_i = u_{i,0}||u_{i,1}||u_{i,2}$*. Let* $u'_i \leftarrow \mathsf{GenTrigger}(u_{i,0}, y_i, K_2, K_3, \{\{A_j, s_j, s'_j\}_{j\in[\ell_0]}\}_{i\in[k]}, i)$.
  Note that the GenTrigger *algorithm is the same as the algorithm in Definition C.6) except taking in the index* i *and using the modified program* Q *in fig. 21.*
- *The challenger flips a coin* b*, and sends* $(\rho_{\mathsf{sk}}, \{u_i\}_{i\in[k]})$ *or* $(\rho_{\mathsf{sk}}, \{u'_i\}_{i\in[k]})$ *to* $\mathcal{A}$*, depending on the outcome.* $\mathcal{A}$ *wins if it guesses* b*.*

```
Hardcoded: $y$; $\{\{A_j, s_j, s'_j\}_{j\in[\ell_0]}\}_{i\in[k]}$; $u_0$.
On input mode, $v$, $i$:

  1. if mode = eval:

     • Choose the $i$-th set of $\{A_j, s_j, s'_j\}_{j\in[\ell_0]}$.
     • Parse $v$ as $v_1, \cdots, v_{\ell_0}$. Output $y$ if and only if for every input $v_j$, if $u_{0,j} = 0$,
       then $v_j$ is in $A_j + s_j$ and otherwise it is in $A_j^\perp + s'_j$.

  2. else if mode = check:
       Parse $v$ as sig$||0^{\ell_0\cdot\lambda-m}$ where the first $m$ bits form a string sig. Check if sig $= y$.
     If so, output 1. Otherwise, output 0.
```

Figure 21: Program Q in k-Hidden Trigger

Proof for Lemma F.3. Similar to the two hidden trigger case of lemma C.7, we can translate the proof for lemma F.1 to the k-wise case naturally. We will refer to each hybrid in the proof for lemma F.1 and point out the differences we make.

We first combine the programs CRSign and CRVK the same way as we did in fig. 17. In hybrid 1, we puncture K_1 on all $u_1, u'_1, \cdots, u_k, u'_k$ and add the check in line 1 for all $\{u_i, u'_i\}_{i\in[k]}$, similar to fig. 18. In hybrid 2, we switch all y_{u_i} from PRF evaluations $F_1(K_1, u_i)$ to uniform random values. In hybrid 3, we puncture K_3 on all $\{u_{i,1}, u'_{i,1}\}_{i\in[k]}$ and add the skip of check for all $u_{i,1}$ or $u'_{i,1}$. In hybrid 4, we puncture K_2 on all $\{u_{i,0}||Q_{u_i}\}_{i\in[k]}$, where Q_{u_i} is an iO of the program Q generated in $\mathsf{GenTrigger}(u_{i,0}, y_i, K_2, K_3, \{\{A_j, s_j, s'_j\}_{j\in[\ell_0]}\}_{i\in[k]}, i)$. In hybrid 5, when running the GenTrigger algorithm for u'_i, we replace the evaluations at punctured points $u_{i,1} = F_2(K_2, u_{i,0}||Q_{u_i})$ with uniformly random value $u'_{i,1}$, for all $i \in [k]$; then in hybrid 6, when running the GenTrigger algorithm for u'_i, we replace $u_{i,2} = F_3(K_3, u_{i,0}) \oplus (u_{i,0}||Q_{u_i})$ uniformly random value $u'_{i,2}$, for all $i \in [k]$.

G Collusion Resistant Copy-Protection for PRFs

G.1 Definitions

Bounded Collusion Resistant Copy-Protection Scheme of PRF A bounded collusion resistant copy-protection Scheme for a PRF family with evaluation algorithm F, consists of the following algorithms:

$\mathsf{Setup}(1^\lambda, k)$: takes in security parameter 1^λ and upper bound k; outputs classical secret key sk;

$\mathsf{QKeyGen}(\mathsf{sk})$: takes in a classical secret key sk; outputs k quantum keys $\rho_{\mathsf{sk}} = \rho_{\mathsf{sk},1} \otimes \rho_{\mathsf{sk},2} \otimes \cdots \otimes \rho_{\mathsf{sk},k}$

$\mathsf{Eval}(\rho_{\mathsf{sk}}, x)$: takes a quantum signing key ρ_{sk} and an input $x \in [N]$; outputs a classical result $y \in [M]$.

A copy-protection for a PRF family with evaluation algorithm $F(\cdot)$ should satisfy the following properties:

Correctness For every polynomial $k(\cdot)$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all λ, all messages m, all $i \in [k]$:

$$\Pr\left[F(\mathsf{sk}, x) = \mathsf{Eval}(\rho_{\mathsf{sk},i}, x) \middle| \begin{array}{c} \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, k), \\ \rho_{\mathsf{sk},1} \otimes \cdots \otimes \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk}), \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Note that the quantum key can be used for polynomially many times, by the gentle measurement lemma [Aar05].

Anti-Piracy Security for k-bounded collusion resistance Let $\lambda \in \mathbb{N}^+$ and F be the evaluation algorithm for a PRF family. Consider the following game between a challenger and an adversary $\mathcal{A}$:

1. The challenger samples $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda, k)$ and $\rho_{\mathsf{sk}} = \rho_{\mathsf{sk},1} \otimes \rho_{\mathsf{sk},2} \otimes \cdots \rho_{\mathsf{sk},k} \leftarrow \mathsf{QKeyGen}(\mathsf{sk})$. It gives ρ_{sk} to $\mathcal{A}$;
2. $\mathcal{A}$ returns to the challenger a (possibly mixed and entangled) state σ on registers R_1, R_2, $\cdots, R_k$. We interpret σ as $k+1$ (possibly entangled) quantum programs $\sigma[R_1], \cdots, \sigma[R_{k+1}]$.
3. The challenger samples uniformly random $x_1, \cdots, x_{k+1} \leftarrow [N]$. Then runs a universal circuit U on2 input $(\sigma[R_i], x_i)$ to obtain y'_i for each $i \in [k+1]$. The outcome of the game is 1 if and only if $F(\mathsf{sk}, x_i) = y'_i$ for all $i \in [k+1]$.

Denote by $\mathsf{CPPRF}(1^\lambda, \mathcal{A})$ a random variable for the output of the game.

We say the scheme has anti-piracy security if for every polynomial-time quantum algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, for all $\lambda \in \mathbb{N}^+$,

$$\Pr\left[b = 1, b \leftarrow \mathsf{CPPRF}(1^\lambda, \mathcal{A})\right] = \mathsf{negl}(\lambda)\,.$$

Intisdinguishability Anti-Piracy Security for PRF The above security defintion is natural but in fact not a meaningful security definition for copy-protecting PRFs considering its cryptographic functionality. Similar definitional issues are discussed in [GKWW21]. We thus provide this stronger definition:

In step 3 of the security game above, the challenger instead does the following:

- It samples uniformly random $x_1, \cdots, x_{k+1} \leftarrow [N]$ as well as random coins $b_1, \cdots b_k \in [k]$. If $b_i = 0$, then it sets $y_i = F(\mathsf{sk}, x_i)$, else it sets $y_i \leftarrow [M]$ to be uniformly random. It then runs a universal circuit U on input $(\sigma[R_i], y_i)$ to obtain a guess b_i' for each $i \in [k+1]$. The outcome of the game is 1 if and only if $b_i' = b_i$ for all $i \in [k+1]$.

Other steps remain the same.

G.2 Construction

The construction for copy-protecting PRFs is the same as the one for signatures(after removing the verification keys and verification programs), since our signing algorithm is essentially a PRF evaluation algorithm.

We formally give the construction as follows. Note that this construction is the same as the one in [CLLZ21], we present the scheme in order to show its extension to collusion resistance.

$\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$:

- Sample PRF keys K_1, K_2, K_3 for F_1, F_2, F_3.
- Output $\mathsf{sk} = (K_1, K_2, K_3)$

$\mathsf{QKeyGen}(\mathsf{sk}) \to \rho_{\mathsf{sk}}$:

- Sample $\{A_i, s_i, s_i'\}_{i\in[\ell]}$: uniformly random subspaces A_i of dimension $\lambda/2$ and vectors s_i, s_i' for $i = 1, 2, \cdots, \ell_0$.
- Let EVAL be the program described in Figure 23. Prepare $\mathsf{iO}(\mathsf{EVAL})$.
- Output the quantum key $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{EVAL}))$,

$\mathsf{Eval}(\rho_{\mathsf{sk}}, x) \to y$:

- Let $\rho_{\mathsf{sk}} = (\{|A_{i,s_i,s_i'}\rangle\}_{i\in[\ell_0]}, \mathsf{iO}(\mathsf{EVAL}))$. Parse x as $x = x_0||x_1||x_2$ where x_0 is of length ℓ_0.
- For all $i \in [\ell_0]$, if $x_{0,i}$ is 1, apply $H^{\otimes n}$ to $|A_{i,s_i,s_i'}\rangle$. Otherwise, leave the state unchanged.
- We obtain state σ from the above procedure (which can be seen as a superposition over tuples of l_0 vectors). Run $\mathsf{iO}(\mathsf{EVAL})$ coherently on input x and σ, and measure the final output register to obtain y.

Figure 22: Quantum copy-protection scheme for PRF key K_1, i.e. functionality $F_1(K_1, \cdot)$.

Correctness and Anti-Piracy Security The correctness and security proofs follow from the proofs for signatures in Appendix C. More specifically, the security proof is simpler because we do not

Hardcoded: Keys K_1, K_2, K_3, R_i^0, R_i^1 for all $i \in [\ell_0]$.
On input $x = x_0||x_1||x_2$ and vectors $v_1, \cdots, v_{\ell_0}$:

```
1. If $F_3(K_3, x_1) \oplus x_2 = x_0'||Q'$ and $x_0 = x_0'$ and $x_1 = F_2(K_2, x_0'||Q')$:
       Hidden Trigger Mode: Treat $Q'$ as a (classical) circuit and output
   $Q'(\mathsf{mode} = \mathsf{eval}, v_1, \cdots, v_{\ell_0})$.
2. Otherwise, check if the following holds: for all $i \in [\ell_0]$, $R_i^{x_{0,i}}(v_i) = 1$ (where $x_{0,i}$ is
   the $i$-th bit of $x_0$).
       Normal Mode: If so, output $F_1(K_1, x)$. Otherwise, output $\perp$.
```

Figure 23: Program EVAL

have a verification program anymore and can thus do reduction to a standard unclonable decryption scheme.

Moreover, the construction satisfies both anti-piracy security notions presented above(the natural definition and the indistinguishability definition), as proved in [CLLZ21] section E.

Remark G.1. *One can observe from the functionality of* EVAL *that PRF key K_1 is the function we copy-protect. We merely use PRF keys K_2, K_3 to assist with the security. However, we sample K_2, K_3 together with K_1 in* Setup*, for the sake of simplicity. If they are sampled independently in each copy of the program, we will result in more redundant notations in the proofs for signatures and hidden triggers, in the k collusion resistant version.*

Therefore, the above construction is the same as the single-copy PRF copy-protection scheme in [CLLZ21] except sampling the auxiliary keys K_2, K_3 in Setup*, not in* QKeyGen*.*

k-Bounded Collusion Resistant Copy-Protection for PRFs The construction is the same as collusion resistant copy-protection for signature scheme in Figure 14. We simply remove the generation of verification key vk and refer the collusion resistant signing program CRSign as EVAL. The correctness and security proofs are straightforward given the proof for the signature case, with modifications to deal with the indistinguishability-based definition as shown in [CLLZ21] section E.